# Path/Action Planning for a Mobile Robot

by

Braden Edward Stenning

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Institute for Aerospace Studies
University of Toronto

# Abstract

Path/Action Planning for a Mobile Robot

Braden Edward Stenning

Doctor of Philosophy

Graduate Department of Institute for Aerospace Studies

University of Toronto

2013

This thesis consists of two major parts united by the common theme of *path/action planning for a mobile robot*. Part I presents the *Second Opinion Planner* (SOP), and Part II presents a new paradigm for navigating, growing, and planning on a *Network of Reusable Paths* (NRP). Path/action planning is common to both parts in that the planning algorithm must choose the terrain assessment or localization technique at the path-planning stage.

There are many terrain-assessment algorithms, and they follow the trend of low-fidelity at low-cost and high-fidelity at high-cost. Using a high-cost, high-fidelity method on all the raw terrain data can drastically increase a robot's total path cost (cost of driving, planning, and doing the terrain assessment). The Second Opinion Planner is a path-planning algorithm that uses a hierarchy of terrain-assessment methods, from low-fidelity to high-fidelity, and seeks to limit high-cost assessment to areas where it is beneficial. The decision to assess some terrain with a higher-fidelity, higher-cost method is made considering potential path benefits and the cost of assessment. SOP provides a means to triage large amounts of terrain data. The system is demonstrated on simulated path-planning problems and in real terrain from an experimental field test carried out on Devon Island, Canada. The SOP plans are quite close to the minimum possible cost.

Growing a network of reusable paths is an approach to navigation that allows a mobile robot to autonomously explore unmapped, GPS-denied environments. This new paradigm results in closer goal acquisition (through reduced localization error) and a

more robust approach to exploration with a mobile robot, when compared to a classic approach to guidance, navigation, and control (GN&C). A NRP is a simple Simultaneous Localization And Mapping (SLAM) system that can be shown to be a physical embodiment of a Rapidly-exploring Random Tree (RRT) planner. Simulation results are presented, as well as the results from two different robotic test systems that were tested in planetary analogue environments.

NRP also offers benefits to planetary exploration missions. NRP allows a rover to be used for the parallel investigation of multiple sites of scientific interest. This dramatically increases the number of sites that can be investigated in a short time, as compared to a serial approach to exploration. Two mock missions were carried out at planetary analogue sites. The first was a purely robotic Lunar sample-return mission, and the second made use of a robotic astronaut-assistant using a network of reusable paths.

# Dedication

This journey was possible because of the continuous support of my family and friends, including those friends who sat with me and advised me. I thank you for your words of encouragement when they were needed, and I thank you for your quiet support when that was needed.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction to Path/Action Planning

Mobile robots have been very successful in Lunar and Mars exploration (e.g., the Mars Exploration Rovers [3, 4, 5, 6, 7, 8, 9], the Sojourner rover [10, 11, 12], and the Lunokhod rovers [13, 14]), and they are expected to continue to aid in our visits to, and investigations of, these and other neighbors in our solar system (e.g., the Mars Science Laboratory [15], a Mars sample-return mission [16]). In order for the next generation of rovers to achieve these objectives, mission controllers must be able to efficiently operate them in natural terrain. Figure 1.1 shows a mobile robot being used during field trials in natural terrain at a planetary analogue site near the Haughton Crater in Nunavut, Canada. Such field tests provide valuable insights into what is necessary for robust and effective planetary exploration, and lessons and feedback from these types of tests have driven the development of the systems and new capabilities that are presented in this work. Many of these capabilities necessary for planetary exploration, and all those presented in what follows, also have applications on Earth. These applications include disaster response [17, 18], mining [19, 20], agriculture [21, 22], transportation [23, 24], logistics [25], security [26], and more.

There are many different types of autonomous capabilities, though primarily this work addresses the capability to do autonomous traversal; the problem of getting from point A to point B in a single command cycle. Previous missions, such as those that

**Figure 1.1:** *A mobile robot being used during field trials in natural terrain at a planetary analogue site near the Haughton Crater in Nunavut, Canada.*

used the Lunokhod rovers in the 1970s [13, 14] (see an example rover in the left image of Figure 1.2), effectively used tele-operation to travel significant distances (Lunokhod 1 traveled 10.5 km in 322 Earth days, and Lunokhod 2 traveled approximately 37 km in about 120 Earth days). However, there are many cases where limited communication windows or communication delays either prevent tele-operation outright, or make the process unusably slow.

Consider the very successful rover-based exploration missions to Mars. The communication delay between Earth and Mars is on average 20 minutes. In 1997, the Sojourner rover [10, 11, 12], as part of the Mars Pathfinder mission, completed short traverses that could be either (i) entirely specified by operators on Earth (i.e., a blind drive), or (ii) defined by setting a nearby goal position for the rover to attempt to reach. The rover could use onboard sensors to detect obstacles in front of itself and it could then turn to avoid and maneuver past these hazards. In total, Sojourner traveled approximately 100 m in 83 Martian days. The Sojourner flight spare is shown in the foreground of the right image of Figure 1.2.

The Mars Exploration Rovers (MERs), Spirit and Opportunity, have capabilities

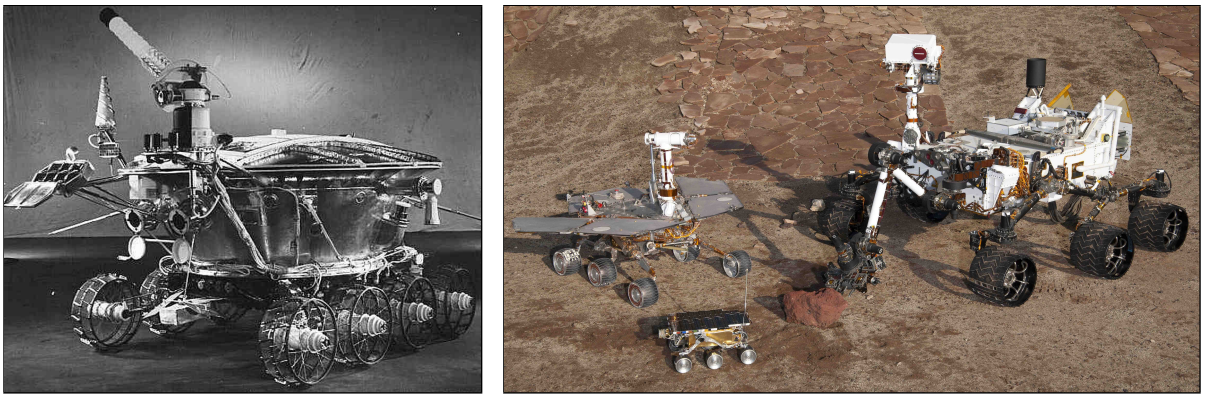

**Figure 1.2:** *Rovers used for Lunar or Mars exploration. The Lunokhod I rover is in the left photograph, and in the right photograph are the flight spare for the Sojourner rover (middle), a Mars Exploration Rover (MER) test rover (left), and a Mars Science Laboratory (MSL) test rover (right). Lunokhod 1, a tele-operated Lunar rover, was 2.3 m in length. Sojourner was 0.65 m in length, the MERs are 1.6 m in length, and the MSL, currently en route to Mars, is 3 m in length. Photo credits: left, Lavochkin Association; right, NASA/JPL-Caltech.*

beyond those of Sojourner. The onboard terrain assessment and path planning allowed them to better detect and plan around nearby obstacles in order to seek autonomously a desired position. A mid-mission software upgrade further improved their autonomous capabilities [9]. Combined, the MERs traveled over 42 km between 2004 and early 2012, and at the time this dissertation was written, Opportunity continued to explore.

The Mars Science Laboratory (MSL) [15], Curiosity, builds upon the heritage of previous missions. MSL arrived on Mars in August of 2012, and it is expected to travel between 5-20 km during its mission. A MSL test rover, the flight spare of the Sojourner rover, and a MER test rover, are shown together in Figure 1.2.

Autonomy is a desirable improvement upon teleoperated systems when it enables more effective or more efficient use of the vehicle, or when it allows the robot to perform tasks that would be otherwise impossible.

There are currently many very capable mechanical chassis, and a great number of different sensors available to a robot. In advanced mobile robots there is typically a sophisticated software system that maps the inputs from these sensors to the commands sent to the vehicle actuators. However, there are no software systems that have been able to exploit robustly and reliably the full capabilities of these vehicles in natural, unrehearsed terrain. For example, the MERs have used short, manual driving commands

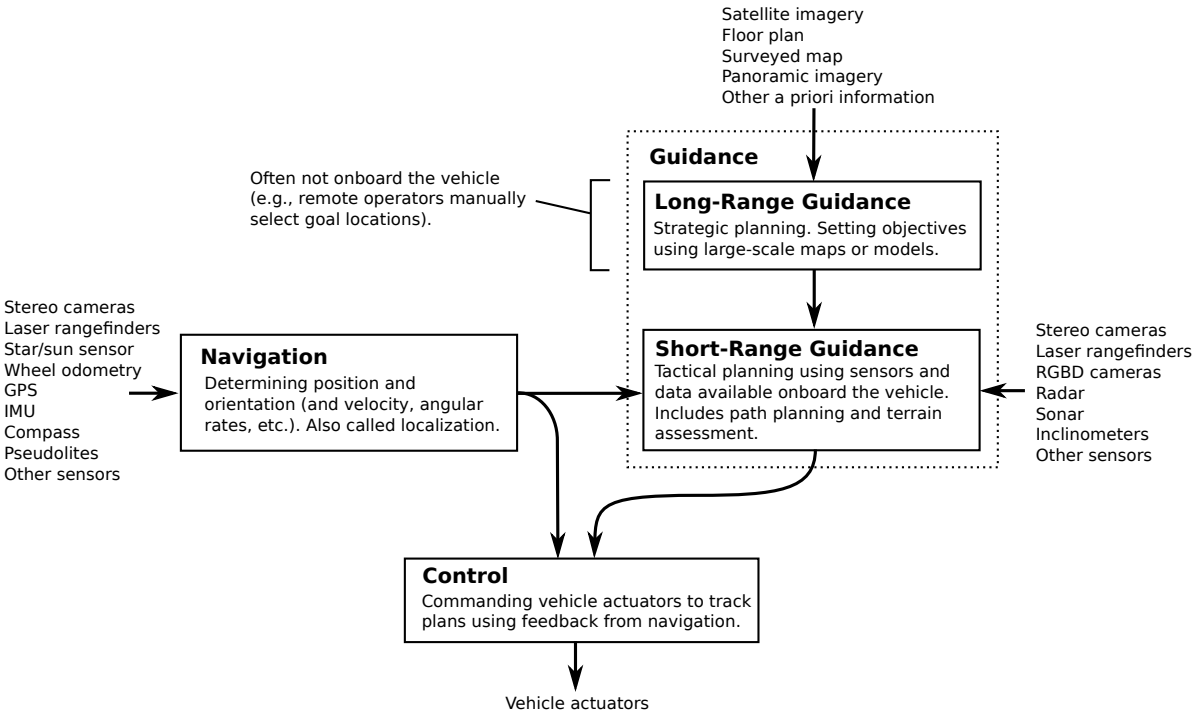


**Figure 1.3:** *A typical guidance, navigation, and control (GN&C) system for a mobile robot. The navigation system uses sensor data to determine the pose of the robot. The pose is then used by the guidance system and the control system. The guidance system sets goal positions and then generates safe plans to those goals using onboard terrain assessment and path planning. The control system then sends commands to the vehicle actuators in order to track those plans according to feedback from the navigation system.*

and limited obstacle avoidance during most of the driving [8]. We can consider this as *the software limiting the autonomy.* Thus, this work seeks to improve the autonomy of mobile robots by improving the onboard software, and we seek to improve the onboard software by improving the underlying algorithms.

The software discussed above, is the guidance, navigation, and control (GN&C) system of an autonomous mobile robot, as shown in Figure 1.3. Each block may contain a set of algorithms, that as a whole, take sensor measurements as inputs and determine appropriate commands to send to the vehicle actuators in order to seek the desired objective.

The navigation system (also known as the localization system) is responsible for determining the pose (position and orientation) of the vehicle relative to some useful reference frame. For example, a terrestrial robot might filter signals from the Global Positioning

System (GPS) as the vehicle moves, and from that it might estimate the vehicle's position and orientation. However, GPS is not available on Mars or the Moon, and even in many situations on Earth (e.g., underground, urban canyons, forests, indoors). Accordingly, other techniques must be used, and there are many techniques from which to choose. Some use dead-reckoning from a single sensor or set of sensors (e.g., visual odometry [27], wheel odometry). Many navigation systems fuse measurements from several sensors in order to improve the localization estimate [28, 29, 30, 31]. Simultaneous Localization And Mapping (SLAM) [32, 33, 34, 35, 36, 37, 38, 39, 40, 41] techniques create and maintain a map of the environment and provide a localization estimate relative to that map. A robot may be able to use a number of localization techniques, and each of these techniques will have its own strengths and weaknesses. For example, there is typically a trade-off between accuracy and computational cost.

The localization estimate is used by both the guidance system and the control system. The control system uses the reference path (e.g., a sequence of poses) or the reference trajectory (e.g., a sequence of pose and time pairs) given in the plan, and the current localization estimate. It then determines the commands to send to the vehicle actuators. This system is responsible for tracking the plan that is given by the short-range guidance system. This action is commonly known as path tracking or trajectory tracking.

The guidance system can be divided into two parts: long-range guidance and short-range guidance. Respectively, these can be thought of as strategic and tactical guidance. The role of the long-range guidance system is to determine the points or poses (commonly referred to as goals or waypoints) that the robot should attempt to reach. The makeup of this system can vary greatly. For example, it may be a large team of people who review many data sources (e.g., orbital imagery or panoramic photos taken onboard the robot) to identify desired scientific targets and set rover poses to allow investigation of those targets. Or, it may be an onboard system that is able to identify targets of interest autonomously [42, 43]. The exact nature of the system is not relevant in the scope of this work, the only requirement is that the long-range guidance system must provide waypoints to the short-range guidance system.

The short-range guidance system then uses this goal along with the current estimated

pose (from the navigation system) to create a safe path plan from the current pose to the goal. The short-range guidance system has two main elements: terrain assessment and path planning. The terrain-assessment system creates a traversability map that, at the most basic level, marks where the robot can and cannot safely drive (more sophisticated maps may try to capture the uncertainty and/or cost of traversal). The path planner then uses this map to plan a safe path from the current pose to the goal. There are many techniques for both terrain assessment [44, 45, 46, 4, 47, 48] and path planning [49, 50, 51, 52, 53]. Many of which are discussed in further detail later in the context of specific parts of this work. Again, these methods each have their own strengths and weaknesses. Terrain assessment, for example, can be conducted using a variety of techniques, and there may be many available to a robot. One of the contributions of Part I of this dissertation is to show that the use of a particular terrain-assessment technique can have a significant impact on the operational efficiency of a mobile robot.

This work is motivated by recognizing that there are many different terrain assessment and localization systems available to an autonomous robot. An individual robot may have a variety of techniques that can be applied to solve each problem. For example, the MERs were manually commanded to use wheel odometry or visual odometry for localization [8]. However, if a robot is able to decide when to use a particular method, the autonomy of the system can be increased, and that means there is an opportunity for the robot to be operated more efficiently. Some previous works have created different operational modes within the GN&C system [54], while others have relied on the path planner to decide when to use the different capabilities [55, 48]. We, too, look to the planner to make these decisions. Thus, not only does the planner have the task of finding a safe and efficient path to the goal, but it must also select the appropriate terrain assessment or localization method. We call this *path/action planning for a mobile robot*.

The remainder of this work is divided into two parts. The parts are united under the theme of path/action planning for a mobile robot, but neither part is exclusively path/action planning. Both parts put a strong emphasis on field tests (test locations are shown in Figure 1.4). Each of the two parts may be read independently of the other. The necessary background for each is included in the appropriate context.

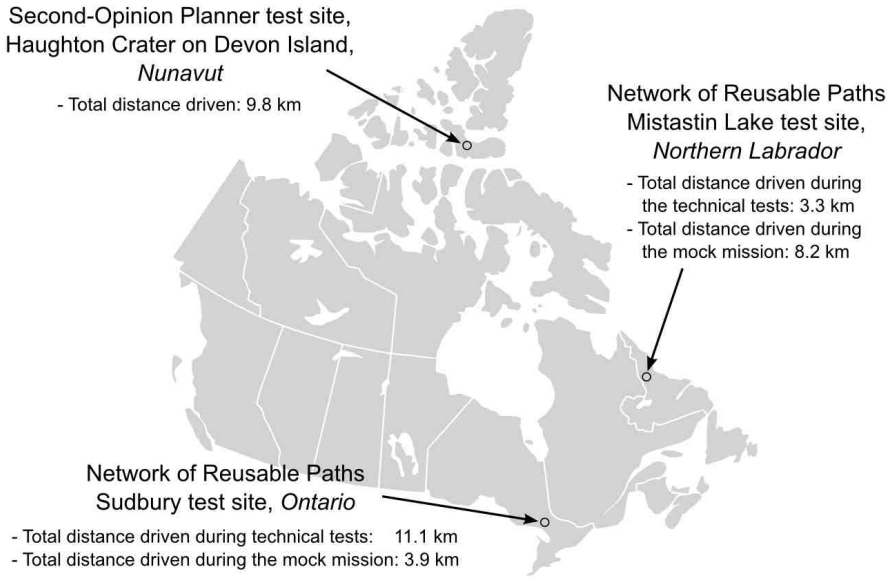


**Figure 1.4:** *Field-test locations, in Canada, of the Second-Opinion Planner and the concept of using a network of reusable paths.*

Part I uses terrain assessment as an action. Chapter 2 begins by recognizing that there are many terrain-assessment algorithms, and they follow the trend of low-fidelity at low-cost, and high-fidelity at high-cost. Using a high-cost, high-fidelity method everywhere can drastically increase a robot's total path cost (the sum of the cost of driving, planning, and doing the terrain assessment). This part presents the *Second-Opinion Planner* (SOP) [56, 57]. The SOP provides an approach to decide which parts of the raw terrain data to process with which assessment methods. It is a path-planning algorithm that uses a hierarchy of terrain-assessment methods, from low-fidelity to high-fidelity, and seeks to limit high-cost assessment to areas where it is beneficial. The decision to assess some terrain with a higher-fidelity, higher-cost method is made considering potential path benefits and the cost of assessment. SOP provides a means to triage large amounts of raw terrain data. The system is demonstrated on simulated path-planning problems and in real terrain from an experimental field test carried out on Devon Island, Canada [58] (see Figure 1.4). In those tests the robot drove approximately 9.8 km. The SOP plans are quite close to the minimum possible cost. Chapter 3 details the system used in the field trials. The three main contributions of Part I are:

1. *the identification of the cost of terrain assessment as an important element in the cost of a path,*

2. *a path-planning framework (SOP) that considers the cost of terrain assessment during path planning, and*
3. *results from field trials of a demonstration system using the SOP framework.*

Part II considers the localization method to be an action, and in Chapter 4, introduces the concept of a *Network of Reusable Paths* (NRP) [59, 60]. NRP is an approach to navigation that allows a mobile robot to autonomously explore unmapped, GPS-denied environments. This new paradigm, when compared to that used a classic GN&C system, results in closer goal acquisition (through reduced localization error) and it is a more robust approach to exploration with a mobile robot. Simulation results are presented along with the results from two different robotic test systems that were tested in planetary analogue environments (Chapter 5). In these two tests, the vehicles traveled a total of more than 14.4 km. The first test was carried out in Sudbury, Ontario, and used a robot equipped with a stereo camera. The second was carried out in the Mistastin impact structure near Mistastin Lake in Northern Labrador. That test used a lighting-invariant system where the robot was equipped with a high-framerate lidar. These test locations are also shown in Figure 1.4. NRP can be thought of as a simple SLAM system that provides many benefits of the more computationally costly approaches to SLAM. The planning algorithm makes NRP into a physical embodiment of the popular Rapidly-exploring Random Tree (RRT) planning algorithm [51, 61]. The three main contributions of Chapters 4 and 5 are:

1. *the development of the concept of a network of reusable paths,*
2. *a path-planning algorithm that uses a network of reusable paths in order to seek distant goals in unknown terrain, and*
3. *results from field tests of two robots using a network of reusable paths to seek distant goals in planetary analogue environments.*

NRP also offers benefits in planetary exploration missions. Chapter 6 shows how NRP allows a rover to be used for the parallel investigation of multiple sites of scientific interest [62, 63]. Parallel exploration can dramatically increase the number of sites that can be investigated in a short time. Two mock missions were carried out at planetary

analogue sites [64, 65, 66]. The first was a purely robotic Lunar sample-return mission where the robot drove approximately 3.9 km. The second benefited from a robotic astronaut-assistant using a network of reusable paths. In the second mission, the rover drove approximately 8.2 km. In Chapter 6, we emphasize three points:

1. *NRP allows a robot to return to a previously visited position with a single command,*
2. *this allows for parallel exploration of scientific sites, and*
3. *parallel exploration allows for an efficient down-selection process to identify key samples for return to Earth.*

This dissertation concludes with Chapter 7. There we finish with final remarks regarding path/action planning, and an overview of the specific contributions that have been made as part of this work.

# Part I

# Terrain Assessment as an Action

# Chapter 2

# Path Planning with Variable-Fidelity Terrain Assessment

The next wave of planetary exploration is going to include rovers that need to travel great distances without constant supervision by Earthbound operators [15]. In the case of Mars exploration, the rover will require greater onboard autonomy that will allow it to move beyond its sensing horizon many times between command cycles from Mission Control [15]. It is therefore imperative to limit the time when the terrain assessment and short-range motion planning are under direct control by operators on Earth.

There may be many terrain-assessment methods available to a mobile robot, ranging from simple methods that can run onboard and in real-time [67], to more complex simulations that include vehicle kinematics and terrain properties [46, 47]. These simulations can often run onboard, but typically not in real-time. Most costly is a call home to ask for human intervention, and this is how much of the Mars Exploration Rovers' (MERs) operations have been carried out [68, 69, 8]. The trend to all these terrain-assessment methods is low fidelity at low cost, and high fidelity at high cost, where we consider fidelity to be a measure of how closely the assessment method is able to model the costs of driving over any patch of ground. The cost can be measured according to any one of many metrics (e.g., time, distance, energy expenditure/gain, risk, wear), but in order to make a meaningful comparison between different paths, one must express all costs in a common set of units. Other requirements may be better handled as constraints. For

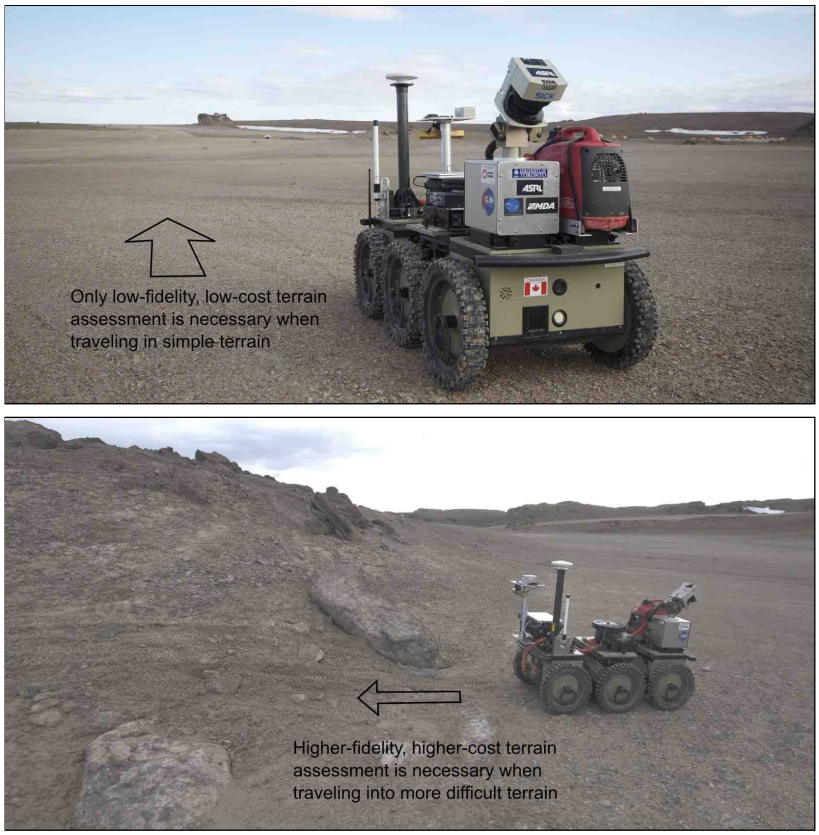


**Figure 2.1:** *Mobile robots can operate in terrain of varying difficulty. Some areas are easily assessed (as in the top image), while others require more advanced methods before the traversability can be determined (bottom image).*

example, if there is insufficient available solar flux, which may be used to generate power, the region is assessed to be an obstacle. In what follows, we choose to use time as the path cost, and compare costs in units of seconds. Different applications may benefit from alternative cost measures.

We can recognize that not all terrain requires a sophisticated assessment to determine traversability. For instance, plane-fit methods, such as those used on the MERs [4], can confidently classify smooth, flat ground as traversable, and tall cliffs as not drivable. This is an example of a low-fidelity assessment method, as it is only accurate in limited types of terrain. The difficulty arises when the rover must decide whether or not it can pass over and into areas that are cluttered or steep; these areas often contain some of the most rich and accessible scientific clues that can be used to improve our knowledge of the site. The challenging regions include areas where geological forces and meteor impacts have

naturally excavated the terrain to reveal many targets of scientific interest.

The next robotic explorers may have many ways of determining terrain traversability, but there will always be a trade-off between the cost of terrain assessment (for instance, computational or communication time) and the fidelity of the assessment method. We can also see that there is a fundamental relationship between path planning and terrain assessment; the path depends on the traversability of the terrain and, ideally, only the regions on the path that will be driven need to be assessed.

In this part, we present a modular path/action-planning framework, the *Second Opinion Planner* (SOP) [56, 57], that uses a suite of terrain-assessment algorithms, limiting the application of the most costly (but most capable) methods only to regions that may be on the optimal path and that require a high-fidelity terrain assessment. This can be thought of as a triage system for the large amounts of raw terrain data that are collected by the robot. Phrased another way, the robot is collecting a large amount of raw terrain data, and SOP provides a way to choose how to optimally process the data. The three main contributions of this part are:

1. *the identification of the cost of terrain assessment as an important element of the cost of a path,*
2. *the development of a path-planning framework that considers the cost of terrain assessment during path planning, and*
3. *results from field trials of a demonstration system using the new framework.*

The work in this part is primarily applicable to slow-moving robots with limited onboard computational resources and limited communication abilities. The example we use are rovers for planetary surface exploration. Indeed, robots that can quickly move beyond their sensing horizon will be more limited by the quantity and quality of terrain data (due to occlusions and limited sensor range) rather than the available onboard processing. Similarly, robots with powerful onboard computers will not have the same practical need to ration their computational power.

This part is organized as follows: Section 2.1 presents some related work, followed by the SOP path/action-planning framework in Section 2.2. In the SOP framework there are two major components: (i) a hierarchy of terrain-assessment methods, and (ii) the

planning framework itself. In Section 2.2.1 we discuss the objective of the assessment hierarchy. Two example hierarchies are presented at later points: (i) a hierarchy for simulated fractal terrain (in Section 2.3.1), and (ii) a proof-of-concept hierarchy developed for, and tested on, real data from field trials on Devon Island, Nunavut, Canada (Chapter 3). In Section 2.2.2 we present the theoretical foundations of the SOP framework before discussing the SOP algorithm itself in Section 2.2.3. Results are presented in Section 2.3. Details of the field system, as well as lessons from the field test, are given in Chapter 3.

## 2.1 Work Related to Path Planning and Terrain Assessment

The path planner has the task of finding an efficient route to a goal location (scientific target, waypoint, etc.), if such a path exists. In order to do this, the planner must know the parts of the surrounding terrain that the robot can pass through, as well as the costs; the planner therefore has an inherent dependence on the terrain-assessment capabilities. In what follows, we will focus on the planning aspect of this problem, acknowledging that there is a variety of appropriate terrain-assessment techniques, but not mentioning specifics other than those used in the test systems.

Some of the graph-based planning techniques are of particular note. A graph is a set of states (nodes) connected by edges. In the typical rover path-planning context, the states represent the locations in the world at which a robot might exist, while the edges contain the associated cost of driving between two adjacent states. The start state corresponds to the current position of the robot, and the goal state is the desired robot position. The A* planning algorithm [49] is an intuitive, optimal planner, and when it is applied to a static, completely known graph, the performance is quite good. However, when the graph is unknown or changing, the A* algorithm is not suitable as it must completely replan after graph updates. The Dynamic A*, or D* algorithm [70, 71], is designed to function efficiently with changing graphs. The algorithms make local repairs to the path as the graph is updated, and thereby maintain an optimal path according to

the known terrain model.

The D* family of algorithms has been extended and used in a variety of applications [72, 73, 74, 75, 76]. A state-lattice path planner, built upon the D* Lite algorithm [52], can ensure that a path is kinematically feasible (i.e., the planned turn radius is greater than or equal to the minimum capable by the chassis). Another such extension is Field D* [77]. Field D* allows for more direct trajectories by planning a path through the edge connecting two states, and then using linear interpolation to determine the path costs. This algorithm has been deployed successfully on the MERs [5, 9]. All these methods require each edge weight to be a non-negative scalar, but the specific method of calculating this weight, often referred to as the edge cost, will depend on the application. Terrain assessment is used to find this edge cost. Some common approaches in robotics measure cost based on time, distance, or energy.

One approach to the problem of automatically switching between two different terrain-assessment methods has been developed in the Terrain Adaptive Navigation (TANav) system [48]. TANav is designed for operation in areas where the robot may experience unpredictable wheel slippage. The initial method of assessment is done in two steps: (i) detect geometric obstacles using a plane-fit technique, and (ii) visual terrain classification that leads to slip prediction. Terrain is identified as definitely traversable, definitely not traversable, or uncertain. The path is planned on a goodness map (where goodness is a measure to quantify the traversability) created by the initial level of assessment, and if the path goes through an uncertain region the second level of assessment, High-Fidelity Traversability Analysis (HFTA), is used to determine the cost of traveling over that section. The cost of using the HTFA is never considered, though it is acknowledged that it is significantly more computationally expensive than the less-capable assessment method. Similarly, Ingrand et al. [54] have developed a framework to improve the autonomy of planetary rovers, allowing the rover to select between two onboard navigation modules (flat or rough terrain navigation), though again, the cost of using the more capable terrain-assessment method is not considered.

Perception-guided path planning (PG2P) [78] is presented as a hybrid approach between navigation (to a desired goal position) and exploration (of the environment). The

intent is to acquire sensor measurements of the most relevant areas when considering how to reach the desired goal position. The system was tested in simulation where it offered improvements in cluttered environments. The algorithm determines where to drive the robot in order to collect the next sensor measurement, but rather than being done in an exploration context, this is done in the context of seeking a goal location beyond the sensing horizon. PG2P deals with data collection, it does not consider the case where there are multiple terrain-assessment methods available to the robot, and that these assessment methods follow the trend of low cost for low-fidelity assessment, and high cost for high-fidelity assessment.

The work of Nabbe and Herbert [55] also deals with planning beyond the sensor horizon or through areas where insufficient data have been collected. This probabilistic framework uses the current map of the environment, the goal location, and the sensing constraints, to find when and where to look in order to maximize the usefulness of potential sensor measurements. When determining a path, the cost of acquiring data is considered and the various outcomes (traversable, obstacle) are simulated using a prediction of the most likely terrain configuration. This prediction is based on the surrounding visible terrain. The system was tested in simulation and in field tests. Like PG2P, this technique provides an answer to the questions of when to look and where to look, *but not which of the available terrain-assessment methods should be used on the large amounts of raw terrain data that are collected*.

In our framework, we consider the cost of terrain assessment (not the cost of perception) during path planning in order to decide which of the available terrain-assessment methods to use (but not where and when to acquire more sensor data). Thus, SOP is carrying out path/action planning with terrain assessment as the action. In practice, SOP could be used in conjunction with one of the above techniques. This would allow the system to decide where and when to collect data, and then SOP would decide which terrain-assessment method should be used with the data.

The PAO* algorithm for planning with hidden state introduces the idea of *pinch points* [79]. Pinch points are areas of uncertain traversability that may have a significant impact on the optimal path. Again, the option of conducting a deeper assessment, with a more

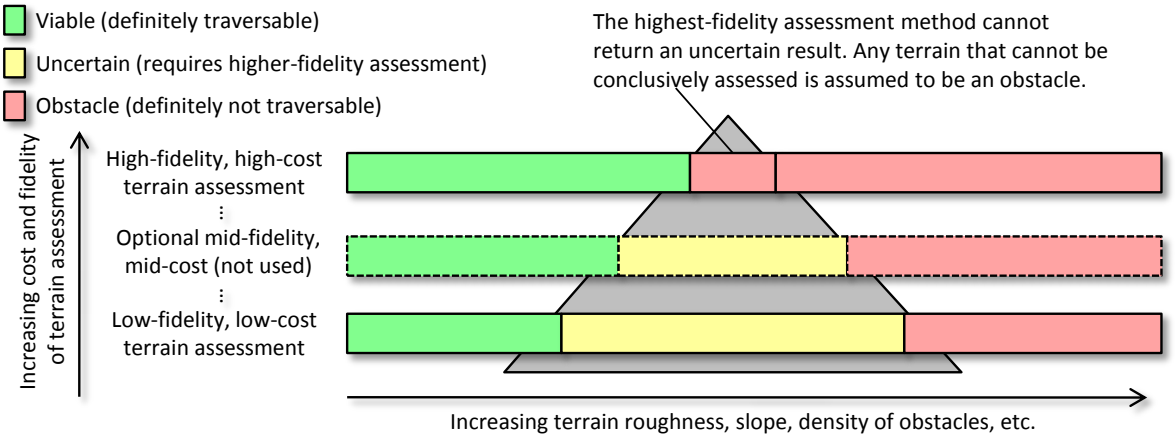


**Figure 2.2:** *A generic terrain-assessment hierarchy with an optional mid-fidelity, mid-cost terrain-assessment method is shown. The cost of using the method increases with fidelity. Increasing fidelity results in conclusive assessment on a greater fraction of all possible terrain (reducing the size of the uncertain region). When not confident of an assessment, a method will return uncertain and a higher-fidelity assessment is needed for a conclusive assessment. In the case of the highest-fidelity method in the hierarchy, any inconclusive results are assumed to be obstacles. All methods are consistent with methods above them in the hierarchy.*

capable assessment method, is not considered. The expectation is that the robot visits the pinch point (or gets close enough to measure it) in order to determine traversability.

To the best of our knowledge, there are no path-planning frameworks, other than the Second Opinion Planner [56, 57], that explicitly account for the cost of terrain assessment during path planning, though there are some that consider the cost of perception [78, 55] during path planning. There are also some that select among the available terrain-assessment methods, but without considering the cost associated with using an assessment method [48, 54].

## 2.2 The Second Opinion Planner (SOP)

This section presents the Second Opinion Planner (SOP) [56, 57]. First, we discuss the hierarchy of terrain-assessment methods that is used in the theoretical foundations of SOP, and then the SOP algorithm itself.

### 2.2.1 A Hierarchy of Terrain-Assessment Methods

We conjecture that it is possible to have a hierarchy of terrain-assessment methods as shown in Figure 2.2. For illustrative purposes we later use two examples of two-level hierarchies, but in practice there can be more than two assessment methods as long as they are ordered by increasing cost and fidelity. The cost must be a non-negative scalar[1]. It is possible that some of these assessment techniques may not actually take place in real-time onboard the robot. For example, manual assessment during a Mars exploration mission might involve having the rover send sensor data to Earth, where operators would make and then return a traversability judgment. This cannot be done in real-time due to the inherent communication delay.

The structure of the hierarchy is such that at the bottom is the low-cost, low-fidelity method. This method uses all the available data and the output must be a cost graph. Conclusive assessment results in an area being *viable* or an *obstacle*. The assessment must be able to mark *uncertain* areas (areas where the method cannot confidently ascertain if the terrain is traversable or an obstacle) and give the corresponding edges in the cost graph a probability of being an obstacle. Some possible approaches to determine this probability are: (i) through empirical data collected at analogue test sites (an example of this method is presented in Chapter 3), or (ii) machine learning techniques (we did not try this).

The high-fidelity assessment method is at the top of the hierarchy, and it is the highest-cost method. The idea is for the planner to prudently use this method only in areas that: (i) are labeled uncertain by the low-fidelity method, and (ii) potentially lie on the optimal path. An important assumption that we make is that the assessment methods are consistent. For example, a high-fidelity assessment on an area deemed traversable by the low-fidelity method must also find the area to be traversable. Additionally, by design, the higher levels in the assessment hierarchy are able to assess a greater fraction of all possible terrain, as shown in the right of Figure 2.2.

[1]If multiple cost metrics are used (e.g., by different assessment methods or as a way to measure the cost of a particular assessment technique), there must be a way to combine them into a single positive scalar. An example would be to use a weighted linear combination of the various metrics.

The fundamental method of assessment can be different at each level in the hierarchy. For example, envision a hierarchy with plane-fit terrain assessment at the bottom. The middle level uses simulation of the vehicle kinematics, dynamics, and local trajectory generation given the current terrain model. The highest level is manual assessment where the robot sends available terrain data and vehicle telemetry (e.g., stereo imagery, point clouds, tilt, power) to human operators who make the traversability judgment. Different levels in the hierarchy may even use different data as part of the assessment process: a stereo camera or a lidar (Light Detection And Ranging), or full field of view data versus high-resolution, targeted photos or scans of an area under consideration.

We can consider the extreme cases of how the two methods may be used. If the high-fidelity terrain assessment is not used on any of the terrain, the result is a long path, as we only allow paths through terrain deemed definitely traversable by the low-fidelity method. If the high-fidelity assessment is used on all the terrain, the robot will spend a huge amount of time processing, and the shorter path will unlikely be worth the effort. Therefore, we require a method to select where to use the high-fidelity terrain assessment. We propose that this decision be made by the planner itself.

### 2.2.2 Theoretical Foundations of the SOP

We have already established that not all terrain need be assessed with the same level of fidelity in order to determine whether or not it can be successfully traversed by a mobile robot. This suggests the use of a framework that judiciously uses expensive techniques. The Second Opinion Planner is a modular path/action-planning framework that uses a hierarchy of terrain-assessment methods and a path planner in an efficient package.

We use a graph-based planning paradigm, where a graph is a set of states, $V$, and edges, $E$. An individual state, $v$, corresponds to a pose in the world where the robot may exist, while an edge, $e$ identifies two states between which the robot can drive, along the associated cost, $w(e)$ (also referred to as the edge weight). An edge can be one of three types: viable, uncertain or obstacle. Viable edges can be traversed for a known cost, and obstacle edges cannot be traversed, giving them an effective weight of infinity. We will omit obstacle edges from graphs and plots. Uncertain edges have an unknown

cost (possibly infinite), but the actual cost may be determined by using a higher-fidelity assessment method. Of course, assessing an edge at higher fidelity incurs an additional cost. The start state corresponds to the current location of the robot and the goal state is the desired location.

SOP attempts to reduce the total path cost by considering the cost of high-fidelity terrain assessment. In order to write this explicitly, let $E_P$ be a path through the graph. $E_P$ is a sequence of connected $n$ edges such that, $E_P = (e_1, ...e_n)$. The cost of assessing the terrain corresponding to an edge is $c_a(e)$. We can then write the total cost of the path as

$$C_p(E_p) = \sum_{i=1}^{n} w(e_i) + \sum_{i=1}^{n} c_a(e_i) + c_{\text{planning}}. \tag{2.1}$$

This is the cost of driving each edge, $\sum_{i=1}^{n} w(e_i)$, the cost of doing the terrain assessment for each edge, $\sum_{i=1}^{n} c_a(e_i)$, and the cost of carrying out the path planning, $c_{\text{planning}}$. Other approaches only consider the cost of driving.

### Determining the Distance Matrices

Consider a high-resolution graph, $G_H := (V_H, E_H)$, that encodes traversability information, for all the visible terrain, based on the low-fidelity terrain-assessment results. The states, $V_H$, and the edges, $E_H$ are defined as

$$V_H := V_L \cup V_{H-L}, \quad \text{and} \quad E_H := E_C \cup E_U, \tag{2.2}$$

where $E_C$ is the set of edges with known weight, $w(e)$ (where $e$ is an individual edge), and $E_U$ is the set of edges with uncertain weight. These disjoint sets contain all the edges in the graph $G_H$. The *lumped state set*, $V_L$, is the set of states consisting of the start state, $v_S$, the goal state, $v_G$, and all states touched by at least one edge in $E_U$. The remaining states, $V_{H-L}$, are the states of $V_H$ that are not in $V_L$ (i.e., those touched only by edges in $E_C$).

In what is to follow, the all-pairs-shortest-paths (APSP) problem is a mental tool useful in the theoretical development of SOP. The APSP problem can be solved using a number of methods [80, 81, 82], but in the implementation of SOP we do not solve the

APSP problem completely, thus we will not elaborate upon these algorithms. The APSP concept allows us to begin developing the theory by considering all paths the robot may take to the goal, and then develop a technique that computes only a smaller set of paths that contain the best path. Consider the APSP solution to $G_H$ stored in the matrix $D_H$, which we will call the *high-resolution distance matrix* (note that in general $D_H$ is not actually completely computed). A distance matrix contains the shortest-path cost between all states, thus, the entry at the $i^{th}$ row of the $j^{th}$ column is the cost of the shortest path from state $v_i$ to state $v_j$. One can also maintain the sequence of edges (i.e., the path from $v_i$ to $v_j$) that are used to calculate the cost, and this path might later be driven by the robot. The operation to create the distance matrix for the graph is represented by a *state-lumping operator*, denoted by $\mathcal{L}$, acting on the graph and the states that are to be included. We introduce another distance matrix, the *lumped-edge distance matrix*, $D_L$, which can be constructed in a similar manner to $D_H$, but starting with $V_L$ instead of $V_H$:

$$D_H := \mathcal{L}(G_H, V_H), \quad \text{and} \quad D_L := \mathcal{L}(G_H, V_L). \tag{2.3}$$

As $V_L \subseteq V_H$, all path weights contained in $D_L$ are also in $D_H$. This means that an equivalent method of finding $D_L$ is to take the relevant information from $D_H$. This can be done by using a projection matrix $P$, which is determined based on the set of lumped states, $V_L$. The $\mathcal{P}$ operator is introduced to represent this operation:

$$D_L = P D_H P^T = \mathcal{P}(D_H). \tag{2.4}$$

The *lumped-edge distance graph*, $G_L$, is just another way of representing $D_L$, where the cost of an edge connecting two states is given by the corresponding entry in $D_L$. For example, the cost of the path that connects state $i$ to state $j$ is the entry at the intersection of the row corresponding to the $i^{th}$ state and the column for the $j^{th}$ state. This is written as $D(i, j)$.

### Assessing Edges of Uncertain Weight

Uncertain edges cannot be traversed and are therefore assigned a naïve weight of infinity, as are edges representing obstacles. However, the weight of uncertain edge may be determined by assessment with a higher-fidelity method. The assessment operator, $\mathcal{A}$, denotes the process of updating a distance matrix by reassessing the edge, $e(v_a, v_b)$, connecting states $v_a$ and $v_b$. Therefore, $D_H$ is updated as

$$D'_H = \mathcal{A}(D_H, e). \tag{2.5}$$

### Updating the Lumped-Edge Distance Matrix

If the size difference between $V_H$ and $V_L$ is large, which it will be if the number of uncertain edges is relatively low, it is highly desirable to be able to update $D_L$ directly. Recomputing and projecting $D_H$, or lumping an updated $G_L$ with the updated $V_L$, are both inefficient and computationally expensive.

**Theorem 2.1.** *$\mathcal{P}$ and $\mathcal{A}$ commute ($\mathcal{P} \circ \mathcal{A} = \mathcal{A} \circ \mathcal{P}$), therefore, $D_L$ can be updated directly.*

*Proof.* The weight being updated, $w'(e)$, by $\mathcal{A}$, always decreases but remains non-negative. The weight is associated with an edge, $e(v_a, v_b)$, that joins states $v_a$ and $v_b$ in $V_L$ where $V_L \subseteq V_H$, so

$$D'_L = \mathcal{A}(D_L, w'(e)) = \mathcal{A}(\mathcal{P}(D_H), w'(e)). \tag{2.6}$$

Let $\delta(v_x, v_y)$ be the weight of the lowest cost path from $v_x$ to $v_y$ for every pair of vertices $(v_x, v_y)$ in $V_L$ so that,

$$\delta(v_x, v_a) + w'(v_a, v_b) + \delta(v_b, v_y) < \delta(v_a, v_b). \tag{2.7}$$

Then update the weight according to,

$$\delta'(v_x, v_y) = \delta(v_x, v_a) + w'(v_a, v_b) + \delta(v_b, v_y). \tag{2.8}$$

This guarantees $D'_L$ represents the true lowest-weight paths. Since taking the alternate

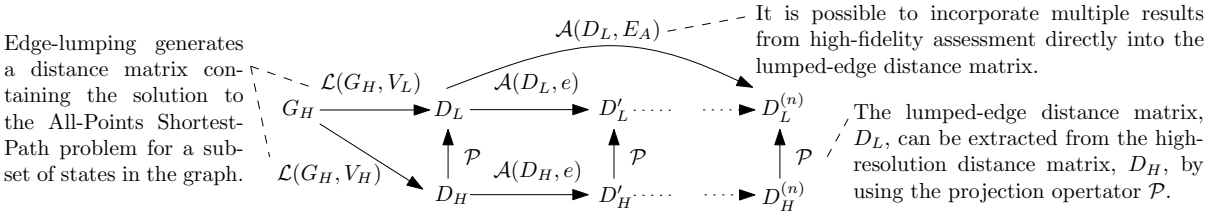


**Figure 2.3:** *A summary of the available operators, their commutativity and the option for batch updates to the lumped-edge distance matrix, $D_L$. The lumping operator, $\mathcal{L}$ operates on the high-resolution graph, $G_H$, to solve the all-points shortest-path problem between a subset of states. In the figure we show lumping with $V_H$ (all the states in $G_H$), and lumping with $V_L$, which is a subset of $V_H$. The assessment operation, $\mathcal{A}$, updates a distance matrix with new edge-assessment results. The projection operation, $\mathcal{P}$, creates $D_L$ from the high-resolution distance matrix, $D_H$. The figure illustrates that we can lump using $V_L$ and perform updates directly on the result, $D_L$. This is equivalent to, and more efficient than, calculating and updating $D_H$.*

approach yields

$$D'_L = \mathcal{P}(\mathcal{A}(D_H, (x, y))), \tag{2.9}$$

and also guarantees that lowest weight paths are represented; $\mathcal{P}$ and $\mathcal{A}$ commute. □

The commutativity can be seen as part of Figure 2.3. It is also desirable to be able to directly update $D_L$ with the results from multiple high-fidelity assessments of uncertain edge weights. Let $E_A$ be the set of $n$ edges, of the form $e(v_a, v_b)$, that are to be reassessed. Since only uncertain edges can be updated, $E_A \subseteq E_U$. Instead of updating $D_L$ with each edge weight individually to get $D_L^{(n)}$, where superscript $(n)$ denotes the $n^{th}$ update, the $\mathcal{A}$ operation can directly produce $D_L^{(n)}$. This leads to the following theorem.

**Theorem 2.2.** *Batch updates to $D_L$ are possible, allowing results of multiple edge assessments to be incorporated simultaneously.*

*Proof.* Consider the $\mathcal{A}$ operator being invoked multiple times where $E_A = \{e_1, e_2, \ldots, e_n\}$:

$$D_L^{(n)} = \mathcal{A}(\cdots \mathcal{A}(\mathcal{A}(D_L, e_1), e_2) \cdots, e_n). \tag{2.10}$$

The update process is extended to handle multiple updates in the following way. Perform individual updates, as before, for all edges in the set $E_A$, and return the final updated distance matrix. The assessment operator, $\mathcal{A}$, then operates on the entire set of edges

and the update to $D_L$ is written as

$$D_L^{(n)} = \mathcal{A}(D_L, E_A). \tag{2.11}$$

Batch updates are then possible because assessments can be carried out one after another without other operations. To simplify notation we write an update as $D'_L$, regardless of the number of edge-weight updates. □

### Selecting Appropriate Edge Weights to Reassess

Potential improvements to the path/action plan can be detected by checking if the total path weight from start to goal is improved if uncertain edges are traversable. Uncertain edges are assigned a heuristic minimum possible cost, $h$ (i.e., $h$ bounds the true cost from below), augmented by the cost of the next-highest-fidelity terrain-assessment method, $c_a$. The distance matrix, $D_L$, which contains the solution to the APSP problem, is then updated using these new weights. If the path weight from start to goal in the updated lumped-edge distance matrix, $D'_L$, is less than that in $D_L$, then the possibility of path improvements cannot be ruled out. This allows us to state the following theorems.

**Theorem 2.3.** *If after reassessment of all uncertain edges contained in the potentially improved path, the total path cost is equal to $D'_L(v_S, v_G) - \sum c_a$ (i.e., all uncertain edges found to have the heuristic cost), then the shortest path from the start to the goal has been found.*

*Proof.* By definition, the heuristic $h(v_a, v_b)$ is an underestimate of the true edge weight, $w(v_a, v_b)$, between states $v_a$ and $v_b$, if such an edge exists. So $h(v_a, v_b) \leq w(v_a, v_b)$. If upon higher-fidelity assessment the true edge weight is equal to the heuristic value, then $h(v_a, v_b) = w(v_a, v_b)$. If by using $h$ the edge is found to lie on the shortest path, then that is the minimum cost path using the actual edge weights since the minimum weights of all uncertain edges were assumed when finding the shortest path. □

The order that uncertain cells are reassessed (if more than one high-fidelity assessment is necessary) influences the number of assessments and therefore the cost of the path and

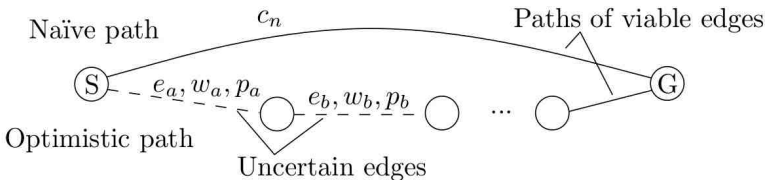


**Figure 2.4:** *Two paths from the start, S, to the goal, G. The naïve path contains only viable edges as determined by the low-cost, low-fidelity terrain assessment. The cost of traveling this path is $c_n$. The optimistic path contains uncertain edges that must be found to be viable according to the high-fidelity assessment before the path can be traversed. The optimistic path may also contain viable edges with a cumulative cost of $c_y$. The uncertain edges, $e_x$, have a heuristic lower bound weight, $w_x$, and a probability of being an obstacle, $p_x$.*

terrain-assessment combination. We can select the appropriate order for uncertain edge assessments by using the following theorem.

**Theorem 2.4.** *Assuming there is a potential path improvement that uses more than one uncertain edge, a minimum average path cost occurs when high-fidelity assessment of $m$ edges is ordered according to*

$$p_1 \geq p_2 \geq \cdots \geq p_{m-1} \geq p_m, \tag{2.12}$$

*where $p_i$ is the probability that the $i^{\text{th}}$ edge that is assessed at the next level of fidelity, is an obstacle.*

*Proof.* Consider two potential paths as shown in Figure 2.4. The *naïve path* only contains viable edges as determined by the low-fidelity terrain assessment. The cost of the naïve path is $c_n$. The *optimistic path* contains $m$ uncertain edges and may contain viable edges. The optimistic path is so named since all uncertain edges must be viable according to the next-highest-fidelity assessment method for the path to be traversable. The cost of driving the optimistic path (not considering the cost of assessment) is $c_p$, and the set of uncertain edges on the optimistic path is $E_R$.

Take any two uncertain edges in the optimistic path, $e_i$ and $e_j$, from $E_R$ and assume all the other uncertain edges in the optimistic path have been assessed as viable. We have a choice: (i) assess $e_i$ first, or (ii) assess $e_j$ first. The average cost of (i) is $c_i$ and

the average cost of (ii) is $c_j$:

$$c_i = c_a + p_i c_n + (1 - p_i)\left(c_a + p_j c_n + (1 - p_j)c_p\right), \tag{2.13}$$

$$c_j = c_a + p_j c_n + (1 - p_j)\left(c_a + p_i c_n + (1 - p_i)c_p\right), \tag{2.14}$$

where $p_i$ and $p_j$ are the respective probabilities that edges $e_i$ and $e_j$ are obstacles. In each case the cost is constructed by considering the cost of terrain assessment at the next level of fidelity and the probability that the naïve path must be taken if one of the assessments finds the terrain to be not traversable. These costs are constructed by assuming the robot must take the naïve path if either of the uncertain edges is an obstacle. Additionally, if the first assessment is an obstacle, the second assessment will not be carried out. Assuming $p_i \geq p_j$,

$$c_i - c_j = c_a(p_j - p_i) \leq 0, \tag{2.15}$$

$$c_i \leq c_j. \tag{2.16}$$

Therefore it is less costly, on average, to assess $e_i$ first and assess $e_j$ later, and only if necessary. Through repeated application of this argument, the assessment order that yields the lowest average cost is,

$$e_1, e_2, \ldots e_{n-1}, e_n, \quad \text{where,} \quad p_1 \geq p_2 \geq \cdots \geq p_{m-1} \geq p_m. \tag{2.17}$$

This concludes the proof. □

Note that this ordering is only carried out when the naïve path is more costly than the optimistic path, including the cost of assessment. Also, if only one uncertain edge is in the optimistic path, then that edge is the first (and only) one to be assessed using the high-fidelity, high-cost terrain assessment. Finally, we can see that if $p_m$ is wrong (perhaps the model of the probability is inaccurate), the system will continue to function, since this property is only used when assessing a set of edges that are part of a potential path improvement, and not in searching for a potential path improvement.

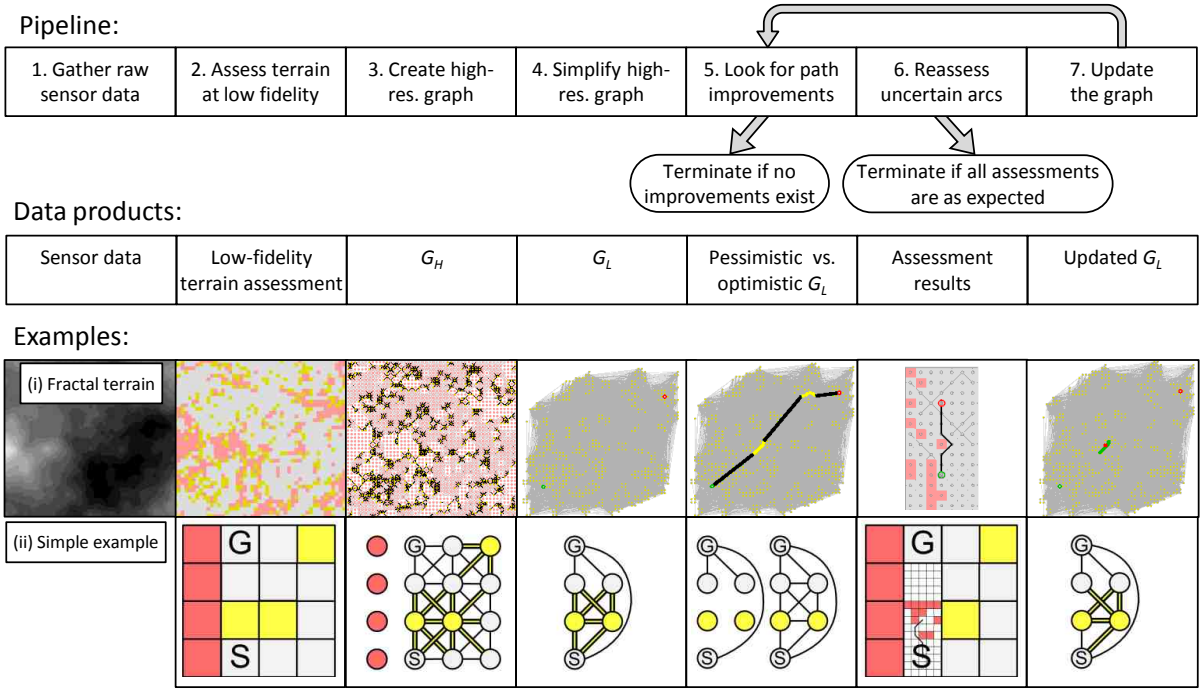


**Figure 2.5:** *The pipeline view of the Second-Opinion Planner algorithm with examples showing the data product at each step: (i) on the top is an example using simulated fractal terrain (using the same terrain map as in Figure 2.6), and (ii) the bottom shows a simple case to better illustrate the steps of the algorithm. The start and goal states, corresponding to the current and desired positions of the robot, are shown as green and red circles or S and G, respectively. Gray areas are viable, yellow areas are uncertain, and red areas are obstacles. Each stage is a major step of the SOP algorithm and the images are representations of the data product created at each step.*

### 2.2.3 The Second-Opinion Planner Algorithm

The SOP algorithm is given in Algorithm 1, and a pipeline view is shown in Figure 2.5. There are two examples: (i) a case using simulated fractal terrain on the top, and (ii) on the bottom, a simple case that more clearly shows the steps. Step 1 of the pipeline is to gather raw sensor data (line 1 in Algorithm 1). Step 2 is to use the data to assess the terrain using the low-fidelity assessment method (line 2 in Algorithm 1). The result is a map where all regions are either known or uncertain. The known areas are either viable (traversable) or an obstacle. Step 3 is to use the low-fidelity assessment results to create a high-resolution cost graph, $G_H$, where edges are viable or uncertain (line 3 in Algorithm 1). The edges have cost of traversal, $w(e)$, which is a function of the traversability criteria used by the assessment technique. Each uncertain edge also has

**Algorithm 1** The Second-Opinion Planner (SOP) algorithm.

1: Gather raw terrain data, $R$
2: Low-fidelity assessment data, $S$, where $S = \text{LowFidAssessment}(R)$
3: $G_H = \text{GraphFromAssessments}(S)$
4: $G_L = \mathcal{L}(D_L)$
5: **while** $G_L$ contains uncertain edges **do**
6: $E_P = \text{FindOptimisticPath}(G_L)$
7: $E_A = \text{ExtractUncertainEdges}(E_P)$
8: **if** $E_A = \{\}$ **then**
9: **return** $E_P$
10: **end if**
11: **for** $e_i$ in $E_A$ **do**
12: **if** $w'(e_i) > h(e_i)$ **then**
13: break
14: **end if**
15: **end for**
16: **if** for each $e_i$ in $E_A$, $w'(e_i) \leq h(e_i)$ **then**
17: **return** $E_P$
18: **end if**
19: $D'_H = \mathcal{A}(D_H, E_A)$
20: **end while**
21: **return** $E_P = \{\}$

probability, $p > 0$, of being an obstacle. For the viable edges, $p = 0$. In this version of the framework, we discretize the terrain as regular, square cells such that $G_H$ is an eight-connected graph. This is an implementation detail, and not a fundamental part of the SOP algorithm. As long as the output of the low-fidelity assessment method is a graph with viable or uncertain edges, the method will work within the SOP framework.

Step 4 is to simplify the high-resolution graph, $G_H$, to create a graph that can be used to both speed up re-planning and to look efficiently for potential path/action improvements (line 4 in Algorithm 1). The simplified graph is the lumped-edge distance graph, $G_L$, and it contains the subset of states connected by paths of uncertain cost, plus the start and goal. The edges between the states of $G_L$ are the edges of uncertain cost and the edges representing the cost to travel between the states along edges of known cost in $G_H$. As part of this step, we find the shortest path from the start to the goal, going through only viable edges and states, we call this the *naïve path*. This path contains

edges $E_P$ and its cost is $c_n$, where,

$$c_n := \sum_{e_i \in E_P} w(e_i). \tag{2.18}$$

A large $G_L$ can greatly increase the computational effort required to run SOP. Therefore, at many points in the algorithm we take steps to keep $G_L$ small. For example, at this point, during the construction of $G_L$, we use $c_n$ as an upper limit when deciding the states and edges to include. The minimum cost of using an edge must be less than $c_n$, and the minimum cost of using an edge from state $v_a$ to $v_b$ is

$$h(v_S, v_a) + h(v_a, v_b) + c_a + h(v_b, v_G), \tag{2.19}$$

where $h(v_a, v_b)$ is the heuristic minimum cost of traversing the edge, and $c_a$ is the cost of terrain assessment at the next level of fidelity. The minimum possible cost of getting to $v_a$ from the start state, $v_S$, is $h(v_S, v_a)$. Similarly, the minimum cost of traveling from the edge to the goal state, $v_G$, is $h(v_b, v_G)$. We use the Euclidean distance between states $v_a$ and $v_b$ for the heuristic distance, $h(v_a, v_b)$. If a state other than the start or goal is not directly connected by an uncertain edge, it is not added to $G_L$. In the tests that were carried out in this part, we have assumed a naïve path always exists. The cost of the naïve path is used as an upper limit of the cost of the paths that are considered in $G_L$, this allows states that are not useful to be left out of $G_L$. If the naïve path does not exist in practice, $G_L$ will contain more states and the computation will be more taxing, but the planner will still find a path, if a path exists. This case is discussed in more detail at the end of this section.

Step 5 has the planner look for path improvements (lines 6 and 7 in Algorithm 1). To understand this step, consider the two extreme cases of what $G_L$ may represent: (i) the *pessimistic* $G_L$, where all the uncertain edges are not traversable, or (ii) the *optimistic* $G_L$, where all the uncertain edges are traversable at the minimum heuristic cost plus the cost of carrying out the terrain assessment at the next-higher-level of fidelity. This allows one to compare the start-to-goal path cost in the optimistic $G_L$ (the optimistic cost) to that in the pessimistic $G_L$ (the pessimistic cost). If the optimistic cost is less than the

pessimistic cost, then there are uncertain edges that may be part of the optimal path and terrain-assessment combination. Stated in an alternative manner: if the start-to-goal path cost changes when the planner is allowed to plan through uncertain regions, there must be uncertain regions that are on the optimistic optimal path. If there are no improvements available, the algorithm terminates (lines 8 to 10 in Algorithm 1).

The pessimistic start-to-goal path cost is available in the pessimistic distance matrix. It is the entry corresponding to the start and goal states, $D_L(v_s, v_g)$. To find the optimistic path, we can use the pessimistic $G_L$ and the uncertain edges to find the start-to-goal entry in the optimistic $D_L$. Recall that $D_L$ is a representation of the graph $G_L$, where the states correspond to the rows and columns of $D_L$, and the cost of the edge connecting state $v_i$ to state $v_j$ is the value of entry $D_L(v_i, v_j)$. The uncertain edges (with edge cost augmented by $c_a$) can be added to the pessimistic $G_L$ and the planner can find a path from $v_s$ to $v_g$ on this new graph. The path is the optimistic start-to-goal path. If this is different than the pessimistic $G_L$ path, there are uncertain edges that could yield path improvements.

Step 6 is to actually reassess the uncertain edges in the optimistic path (in order of decreasing $p$) using the terrain-assessment method at the next-higher-level of fidelity. The result is that we now know the actual cost of the reassessed edge. If the reassessment result is that the edge is traversable at the heuristic cost (as hoped for in the optimistic $D_L$), then the shortest path has been found. If an edge is assessed as an obstacle, the reassessment is immediately stopped, even if there are other assessment requests pending, since the optimistic path is now not traversable. Figure 2.5 shows a high-fidelity assessment that uses a higher-resolution map with local planning, this is an example of one method of high-fidelity assessment that may be used. In Algorithm 1 Step 6 is shown in lines 11 to 18.

When necessary, step 7 is executed to update the graph $G_L$ with the traversability costs of the reassessed edges (line 19 in Algorithm 1). We then return to step 5 and the planner can again look for further improvements. The planner will continue to seek a second opinion on pieces of terrain until the assessment costs outweigh the possible path benefit or until no uncertain edges remain (line 5 in Algorithm 1).

Let us return to the case where the naïve path does not exist. In our implementation,

the planner will always find a path, if one exists. This is guaranteed by the algorithms used in the implementation and remains true even when there is no naïve path. The case where no naïve path is present is easy to detect, and therefore could allow one to use a different architecture (i.e., not SOP) when this is the case. However, one could also just allow SOP to continue. Consider the case where all the terrain is marked uncertain by the low-fidelity method (i.e., no obstacles, no viable sections), the search space for the high-fidelity assessments will include all the connected, uncertain edges. Therefore, the planner will continue requesting high-fidelity terrain assessments until either a viable path is found, or all possibilities have been exhausted and no path is found. From a different perspective, the naïve path is simply the backup plan, and the heuristic based on the cost of this backup plan indicates when the planner should give up on more high-fidelity assessments and just use the naïve path.

## 2.3 Results of the Second-Opinion Planner

The Second-Opinion Planner has been tested on both simulated fractal terrain, and using real terrain data collected at a planetary analogue site on Devon Island, Nunavut, Canada (the location of the test site is shown in Figure 1.4). The results are presented here. In both of these tests, the planner had access to a large amount of raw terrain data that could be used for terrain assessment, but the cost of processing all the data at the highest-fidelity (and highest-cost) would have been prohibitive. SOP provided a method to select which assessment method to use with which data.

### 2.3.1 Tests of SOP on Simulated Fractal Terrain

We generated fractal terrain [83] (a planner test method similar to that of Stentz [70]) to test the performance of the algorithm and to compare it to other methods. Each terrain model is $1008 \times 1008$ pixels, with each pixel value encoding to the local terrain height (as a floating-point number). The low-fidelity terrain-assessment method divided the terrain model into coarse, square cells that are 21 pixels per side. The traversability of a cell was estimated based on the roughness (variance of the height) of the pixels in the cell. There

were two roughness thresholds: (i) a limit below which a cell was definitely traversable, and (ii) a limit above which the cell was an obstacle. If the roughness was between the two thresholds, the cell was deemed uncertain. The states in the high-resolution graph corresponded to the centers of the cells used for the low-fidelity assessment. In the test system, the edges were based on an 8-connected graph. These edges were added if the associated states were not obstacles, and if the difference between the mean heights was below the maximum step threshold. If both cells were viable, the edge was viable; otherwise, the edge was marked as uncertain. The probability that an edge was an obstacle was modeled using a function based on the roughness (similar to the function used in Chapter 3).

The high-fidelity assessment operated on an uncertain edge in the high-resolution graph. The pixels associated with the edge were grouped into square cells of three pixels per side. This means the coarse cells used in the low-fidelity assessment were broken into a grid of $7 \times 7$ smaller cells. A local cost graph was created using the same method as the low-fidelity assessment, except there was no uncertain option, since the high-fidelity method was at the top of the assessment hierarchy. On the local cost graph, we searched for a path from the tail to the head of the uncertain edge. Superficially, this may appear similar to multi-resolution path-planning techniques [84, 85, 86]. However, this is purely a result of the choice for the high-fidelity method in the assessment hierarchy. SOP is able to use a variety of high-fidelity assessment methods, and these do not necessarily need to be higher-resolution versions of the low-fidelity assessment methods. In this section, one should take the example high-fidelity assessment as a placeholder for some arbitrary method with a given cost of assessment. With this, one can predict the performance of an assessment hierarchy that has similar scaling between the underlying costs.

Figure 2.6 shows a low-fidelity assessment with a detailed look at two examples of high-fidelity assessment. A SOP path plan is also shown. It is an improvement over the naïve path plan. The SOP plan went through uncertain cells that required assessment using the high-fidelity terrain-assessment method.

In our tests we positioned the start and goal in locations near to those shown in the example, allowing slight adjustments if the start or goal was situated on an obstacle or

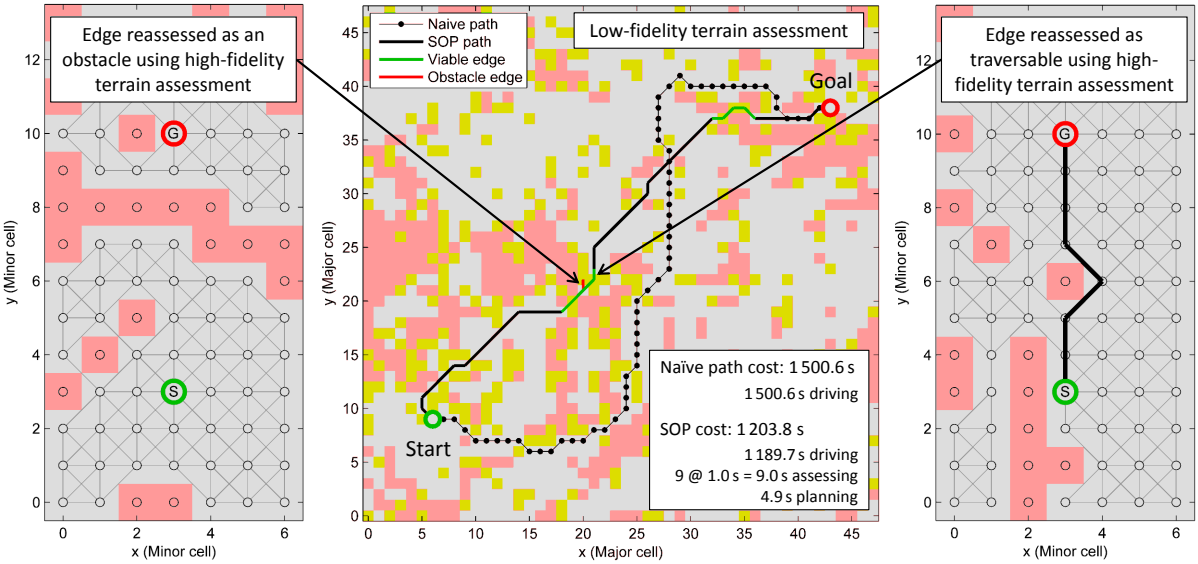


**Figure 2.6:** *A sample SOP plan on simulated fractal terrain. In the center is the low-fidelity terrain assessment with the resulting paths and high-fidelity terrain-assessment locations. The naïve path (dashed) goes through only viable regions (grey) while the SOP path (solid) crosses through areas that were uncertain (yellow) according to the low-fidelity assessment. High-fidelity assessments of edges that were found to be obstacles are red while viable edges are green. Sample high-fidelity assessments of an obstacle and traversable edge are shown on the left and right, respectively. Note that a single cell in the low-fidelity terrain assessment corresponds to seven cells in the high-fidelity assessment.*

uncertain cell. Additionally, if there was not a naïve path (i.e., all certainly traversable edges) from the start to the goal, the map was not used. We ran the SOP algorithm on 1089 fractal terrain maps and compared the results with the alternative options that are described later. The results are summarized in Figure 2.7 and Table 2.1. Table 2.1 provides the average and standard deviation of the costs across all the maps for three different values for the cost of high-fidelity terrain assessment. The total cost is broken down into the three main components: (i) path length, (ii) number of high-fidelity assessments, and (iii) planning time. We also provide a total-time cost that combines the three components by assuming a driving speed and time for each high-fidelity assessment. The contributions to the total cost must all be expressed in the same units. In our examples we have used units of seconds. The total cost is then just a weighted linear combination of the distance, number of assessments, and the planning time, as seen in Table 2.1. The speed of the robot was 1 pixel/s, and three cases are considered with different costs of assessment, $c_a$: (1) 0.03 seconds/assessment, (2) 3.5 second/assessment,

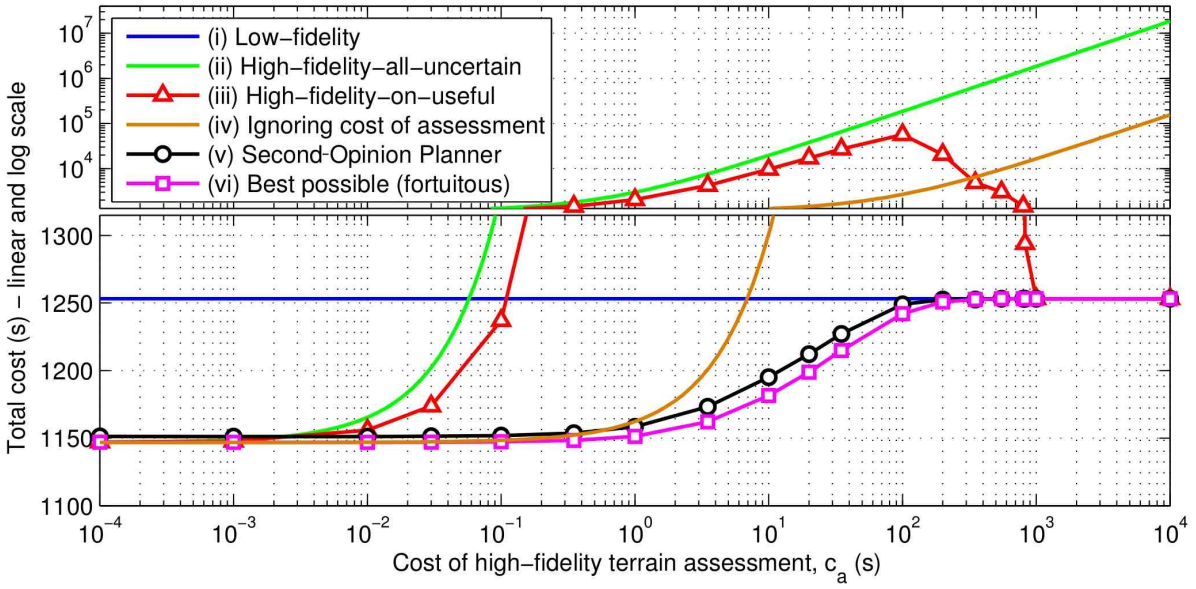


**Figure 2.7:** *Total cost using different values for the cost of high-fidelity terrain assessment, $c_a$. The total cost is the combination of the time to drive the path, carry out any high-fidelity terrain assessments, and do any planning operations. The methods considered are: (i) low-fidelity assessment used everywhere, (ii) high-fidelity assessment used on all uncertain edges, (iii) high-fidelity assessment applied to the uncertain edges in $G_L$, (iv) the cost of assessment is not considered during planning, (v) the Second-Opinion Planner, and (vi) the best possible case which is a fortuitous lower bound created if high-fidelity assessment is only carried out on uncertain edges on the final path (this is not achievable in practice because the final path is not known in advance). Cases (i), (ii) and (iv) can be calculated for any $c_a$ but the other methods require simulation; thus the markers correspond to results from a batch of simulations at a particular value of $c_a$. In all,* 19 *values of $c_a$ were used on* 1,089 *fractal terrain maps for a total of* 20 691 *tests.*

and (3) 100 seconds/assessment, and we can see that this cost had a dramatic influence on the choices made by the SOP algorithm. The tests were carried out on one core of an Intel®Core™2 Duo 2.4 GHz processor with 3 GB of RAM. The average distance between the start and the goal is 959 pixels with a standard deviation of 32.6 pixels.

The conditions for the simulations could represent many possible robot systems depending on the scale of the terrain pixels. On the relatively large scale, we may have a robot that uses a coarse resolution for terrain assessment (e.g., 1 m) and therefore travels relatively quickly, given the scaling (i.e., 1 pixel/s), this leads to a ground-speed of 1 m/s. This would mean the goal was approximately 959 m from the start position. For a smaller scale, consider a robot that uses a finer resolution for terrain assessment, 0.05 m cell resolution, and that therefore travels 0.05 m/s. With this scaling, the goal would be

**Table 2.1:** *Results for three cases corresponding to three different values of the cost of assessment, $c_a$. Case 1, 2 and 3 use high-fidelity assessment costs of 0.03, 3.5 and 100 seconds/assessment respectively. The methods are as in Figure 2.7. This table presents the average cost of each contribution and case over all 1 089 fractal terrain maps and gives one standard deviation in parentheses. The value from the lowest-cost planner in each case (excluding method (vi), which is not possible a priori) is shown in bold.*

| | Path Length (pixels) | High-Fid. Assess. (#) | Planning Time (s) | Cost of Case 1 (s) | Cost of Case 2 (s) | Cost of Case 3 (s) |
|---|---|---|---|---|---|---|
| (i) Low-fidelity | 1 253.23 (±177.3) | | 0.029 (±0.0079) | 1 253.2 (±177.3) | 1 253.2 (±177.3) | 1 253.2 (±177.3) |
| (ii) High-fidelity-all-uncertain | 1 146.9 (±111.3) | 1 843.9 (±118.9) | 0.031 (±0.0091) | 1 202.2 (±110.9) | 7 600.7 (±419.0) | 185 540 (±11 876) |
| (iii) High-fidelity-on-useful | | | | | | |
| Case 1: $c_a = 0.03$ s | 1 146.9 (±111.3) | 893.0 (±459.8) | 0.046 (±0.012) | 1 173.9 (±120.4) | | |
| Case 2: $c_a = 3.5$ s | 1 146.9 (±111.3) | 876.0 (±468.9) | 0.046 (±0.013) | | 4 228.1 (±1 717.1) | |
| Case 3: $c_a = 100$ s | 1 146.9 (±111.3) | 556.0 (±639.8) | 0.045 (±0.013) | | | 56 843 (±64 074) |
| (iv) Ignoring cost of assessment | 1 146.9 (±111.3) | 15.4 (±12.3) | 0.031 (±0.0091) | **1 147.4** (±**111.5**) | 1 200.8 (±135.0) | 2 685.9 (±1 279.3) |
| (v) Second-Opinion Planner | | | | | | |
| Case 1: $c_a = 0.03$ s | 1 146.9 (±111.3) | 8.7 (±7.2) | 4.7 (±4.6) | 1 151.5 (±113.5) | | |
| Case 2: $c_a = 3.5$ s | 1 147.0 (±111.4) | 7.1 (±5.6) | 4.6 (±4.6) | | **1 173.2** (±**120.1**) | |
| Case 3: $c_a = 100$ s | 1 220.8 (±142.2) | 0.29 (±0.94) | 2.2 (±3.4) | | | **1 249.1** (±**169.7**) |
| (vi) Best possible | | | | | | |
| Case 1: $c_a = 0.03$ s | 1 146.9 (±111.3) | 4.8 (±2.9) | | 1 147.1 (±111.3) | | |
| Case 2: $c_a = 3.5$ s | 1 147.0 (±111.4) | 4.4 (±2.8) | | | 1 162.1 (±113.3) | |
| Case 3: $c_a = 100$ s | 1 220.8 (±142.2) | 0.21 (±0.69) | | | | 1 242.1 (±152.9) |

approximately 47.95 m away. These distances correspond to the common ranges available for lidar (LIght Detection and Ranging) sensors.

In Figure 2.7 and Table 2.1, the *low-fidelity* assessment case employed the low-fidelity assessment method everywhere and planned on the resulting graph without allowing the path to traverse uncertain edges (i.e., the path was the naïve path). As we would expect, the path length was relatively long, and the path length was the major factor in the total cost. The *high-fidelity-all-uncertain* assessment case began with the same graph, but then used the high-fidelity assessment method on all the uncertain edges. The result was that the path taken was the shortest possible path, but there was a huge cost associated with carrying out all the high-fidelity assessments. In the *high-fidelity-on-useful* case, the high-fidelity assessment was only carried out on the smaller set of uncertain edges

included in $G_L$. The result was again the shortest possible path, but with fewer high-fidelity assessments than the high-fidelity-all-uncertain case. The number of assessments varied with $c_a$ because $c_a$ was a factor in the selection of the states and edges to include in $D_L$. For *ignoring the cost of assessment*, the cost of high-fidelity terrain assessment was neglected during planning. Therefore the planner would plan along uncertain edges if that was the shortest path. There was no consideration for the eventual cost of assessment that would be incurred; this method is most similar to the TANav framework [48] mentioned previously. In all these cases the cost of planning was negligible.

SOP, on average, found the lowest-cost combination of path and terrain assessment, or it was quite close for all cases. It was only slightly more costly at low $c_a$, where the planning overhead increased the time taken by 2 to 6 seconds. Of course, if the high-fidelity terrain assessment were this cheap it would have been used as the low-fidelity method, allowing for even more capable methods higher in the assessment hierarchy. The SOP costs were 1151.5 s, 1173.2 s, and 1249.1 s, this corresponds to improvements of 101.7 s (8.12%), 80.0 s (6.38%), and 4.1 s (0.33%) for cases 1, 2 and 3 respectively. SOP used the shortest path most of the time and used very few high-fidelity assessments, though clearly to good effect. The cost of planning was small but not insignificant. SOP offered an advantage over the other options shown, finding what was quite close to the minimum possible path/assessment combination. The *best possible* path/assessment combination would have been to only assess uncertain edges that were on the final path, but it was impossible to know in advance if an edge would turn out to be an obstacle or traversable after high-fidelity assessment. The best-possible curve is shown and it acts as a theoretic lower bound of achievable performance.

From Figure 2.7 we can see that there was a range of $c_a$ where SOP clearly outperformed the other methods. The width of this range depends on the sensor range (the size of the map), since as the map gets bigger there would be cases where even higher-cost high-fidelity methods may be used wisely. Additionally, the curve depends on the speed and computational resources of the robot. Further important relationships may also arise when considering the actual robot system. This implies that the performance can benefit from good design, though SOP can offer improvements even if the choice of

terrain-assessment methods is constrained.

Looking critically at the results, there are some opportunities for improvement. The planning time can be reduced with a more efficient implementation, and it is dependent on the computing resources. The number of assessments may be also be reduced by moving away from an optimistic model and making better use of the probability that an edge is an obstacle. In fact, the desire for this ability strongly influenced the development of the fundamental SOP elements, namely, the use of the distance matrices, which allow the system to retain the ability to consider many alternatives at once. In contrast, the path length cannot be appreciably reduced since it is already quite close to the theoretical minimum. Further improvements are going to be relatively small, and in practice, only noticeable on large maps where the cost of high-fidelity terrain assessment is quite significant relative to the driving speed.

### 2.3.2 Real-World Results

SOP has also been used on real-world data with a proof-of-concept terrain-assessment hierarchy. The hierarchy is described in Chapter 3 and the results are shown here. The planning tests were run offline using the data collected in the field. The data collection, using the robot shown in Figure 2.1, was part of field trials held in July 2009 on Devon Island, Nunavut, Canada. For a primary guidance sensor, the robot was equipped with a forward-looking planar laser rangefinder (SICK) with a 180° field of view, mounted on a pan-tilt unit (PTU). Localization was provided using visual odometry from a rear-facing stereo camera. A differential GPS (DGPS) was used to record ground-truth position. Inclinometers in the front and rear body sections provided roll and pitch measurements for the SICK and PTU, and for the DGPS antenna. We also took long-range lidar scans (with a range of a couple hundred meters to over a kilometer) of the traverse site using an Optech ILRIS-$3_6$D. The driving data from the robot, along with the long-range scans of the traverse site, were used to create an assessment hierarchy that used the long-range lidar data. This allowed for SOP to be used to plan paths on other long-range scans, and those results are shown here. Further descriptions of the robot and the terrain-assessment hierarchy are presented in Chapter 3.

**Table 2.2:** *Planning results using 10 long-range scans collected from Devon Island, Nunavut, Canada. Comparison planners are as described in Figure 2.7. The value from the lowest-cost planners (excluding method (vi), which is not possible a priori) is shown in bold for each test.*

| Scan | Start-to-goal distance (m) | (i) Low-fid. path cost (s) | (ii) High-fid.-all-uncertain path cost (s) | (iii) High-fid.-on-useful path cost (s) | (iv) Ignoring-cost-of-assess. path cost (s) | (v) SOP path cost (s) | (vi) Best possible path cost (s) |
|---|---|---|---|---|---|---|---|
| 1 | 272.0 | 931.6 | 55 725.9 | 51 674.7 | 803.5 | **749.6** | 722.3 |
| 2 | 169.7 | 405.7 | 7 619.7 | 13 147.9 | 383.3 | **382.4** | 375.6 |
| 3 | 128.1 | 619.1 | 17 516.1 | 32 708.1 | 594.8 | **554.4** | 551.7 |
| 4 | 210.0 | 459.7 | 15 220.6 | 26 466.9 | 535.9 | **439.9** | 429.5 |
| 5 | 570.1 | **1 122.4** | 54 146.9 | 77 504.6 | 1 313.4 | 1 151.0 | 1 100.4 |
| 6 | 344.4 | 818.0 | 32 923.7 | 47 567.1 | 768.2 | **691.6** | 661.8 |
| 7 | 116.6 | 460.3 | 82 184.3 | 35 194.6 | **346.3** | 347.5 | 338.3 |
| 8 | 164.0 | 518.2 | 35 764.4 | 26 037.3 | 486.1 | **471.8** | 469.6 |
| 9 | 240.2 | 741.2 | 29 322.0 | 49 641.3 | 837.9 | **628.9** | 603.9 |
| 10 | 216.3 | 1 114.0 | 29 841.6 | 58 432.1 | 1 420.8 | **1 108.0** | 1 106.5 |

Table 2.2 contains the results from 10 distinct long-range lidar scans. The start and goal locations were selected so that there was a naïve path and potential for improvements that go through uncertain regions. The methods are as defined in Figure 2.7 with the cost of high-fidelity terrain assessment set to 20 s/assessment.

Figure 2.8 presents a more detailed look the results from scan 1. Here we contrast the results of SOP (right) with neglecting the cost of high-fidelity terrain assessment at the planning stage (left). In this case, SOP did not select the shortest driving distance path, but instead, a compromise between driving time and time spent carrying out high-fidelity terrain assessment. SOP found a shorter path (according to total time) than any other method, including the naïve path, where only low-fidelity terrain assessment is used. Figure 2.9 shows two more SOP plans (scan 7 on the left and scan 5 on the right) using data collected during field trials.

SOP is not guaranteed to find the lowest cost path in any particular case (only on average). In certain cases (such as scan 5) the SOP cost can be higher than the naïve path. This situation typically arises when a great number of high-fidelity assessments return as obstacles. Over the course of these 10 tests, on average, SOP found plans that were 89.78% the cost of the naïve path, and 89.65% the cost of ignoring cost of assessment during planning. SOP offers improvements of approximately 10% over the other methods considered. Also note that, on average over these tests, using high-fidelity methods while ignoring the associated high cost is more costly than solely using the low-cost, low-fidelity

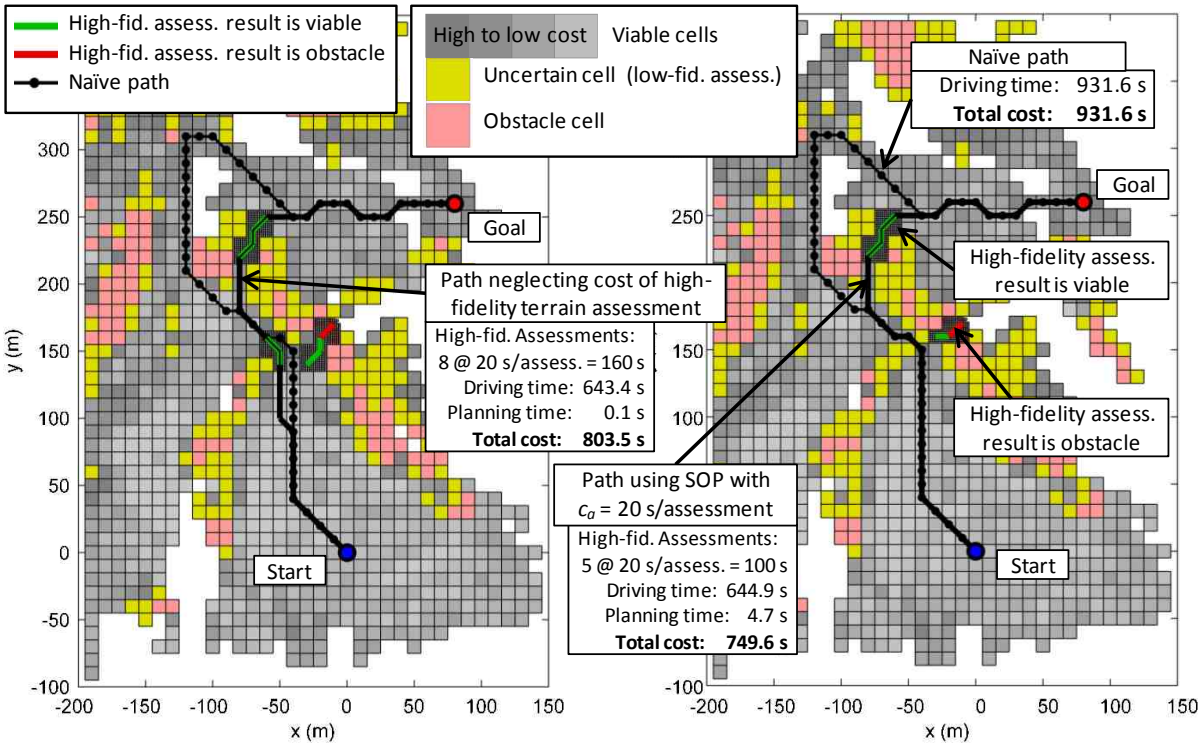


**Figure 2.8:** *A detailed look at the results from scan 1. On the left is the result if the cost of high-fidelity terrain assessment ($c_a = 20$ s/assessment) is ignored during planning. On the right is the SOP plan on the same terrain. In both cases the path cost was lower than that of the naïve path. However, even though the driving time on the left was less, the total time was greater than the total SOP time.*

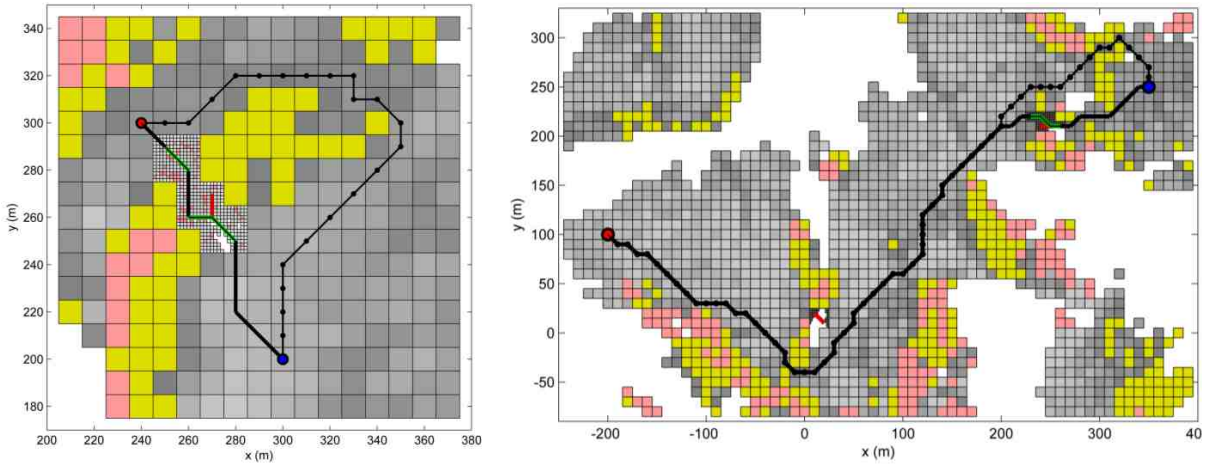


**Figure 2.9:** *Additional SOP results for scan 7 (left) and scan 5 (right). The parameters and the legend are the same as in Figure 2.8. In scan 7 the SOP path cost was 347.4 s compared to the low-fidelity path cost of 460.3 s. In scan 5 those same costs were 1 151.0 s and 1122.4 s, respectively.*

terrain assessment and driving longer paths. It is important to account for the cost of terrain assessment in cases where use of high-fidelity assessment is frequent, or the cost becomes significant. We would expect that with a different hierarchy, more tailored to the particular system (as predicted by Figure 2.7), the improvement over neglecting the cost of terrain assessment would be more pronounced.

## 2.4 Conclusions

The Second-Opinion Planner (SOP) is a path/action-planning framework that considers the cost of terrain assessment in the planning process. It attempts to limit the use of high-cost, high-fidelity terrain assessments to areas that can result in a lower cost path (including the cost of assessment). SOP provides a means to triage the large amounts of raw terrain data that are collected by the robot. In this chapter, we presented the motivation for SOP, the SOP algorithm proper, and the results of SOP field tests. More specifically the three main contributions of this part are:

- *the identification of the cost of terrain assessment as an important element in the cost of a path,*
- *the SOP path/action-planning framework which considers the cost of terrain assessment during path planning, and*
- *results from field trials of a demonstration system using the SOP framework.*

In support of these contributions, we have presented the theory behind the framework and the results of using the Second-Opinion Planner framework on both simulated fractal terrain and real-world data collected at a Mars analogue site on Devon Island, Nunavut, Canada. Included in this, we demonstrated a proof-of-concept implementation of a two-level terrain-assessment hierarchy using real sensor data, and verified that the assumptions in this hierarchy are plausible. While some further improvements are possible, the SOP plans are already quite close to the minimum possible cost. Details of the development of the real-world assessment hierarchy are given in Chapter 3.

# Chapter 3

# A Real-World Terrain-Assessment Hierarchy

A primary goal of the field trials was to show, by creating a real-world example, that our assumptions on the existence of a terrain-assessment hierarchy are reasonable. In order to do this, we collected performance data (speed, number of stops) for the robot, shown in Figure 2.1, traveling through a variety of terrain at a planetary analogue site near the Haughton Mars Project base camp (near Haughton Crater) on Devon Island, Nunavut, Canada. We also took long-range lidar scans of these areas using an Optech ILRIS-$3_6$D.

The onboard short-range guidance system allowed the robot to travel hundreds of meters and avoid local obstacles. The guidance system operated in two main modes. In mode one (*push-broom*), the robot would drive straight at the goal with the SICK laser rangefinder angled forward and down to look a short distance ahead ($0.5 - 2\,\mathrm{m}$ depending on the robot's speed) to detect hazards. If a hazard was detected, the robot would slow down, and if necessary, stop. If a stop was required, the robot would reverse along its track a short distance. It was necessary for the vehicle to back up because the obstacle detection range was short and the vehicle could only make large radius turns. When an obstacle was detected directly in front of the vehicle, it needed to back off in order to get the space necessary for an avoidance maneuver. We chose to have the vehicle back up along its inbound path because there was no rear-facing hazard detection and we knew the path was safe since the robot just drove it. Note that this realization

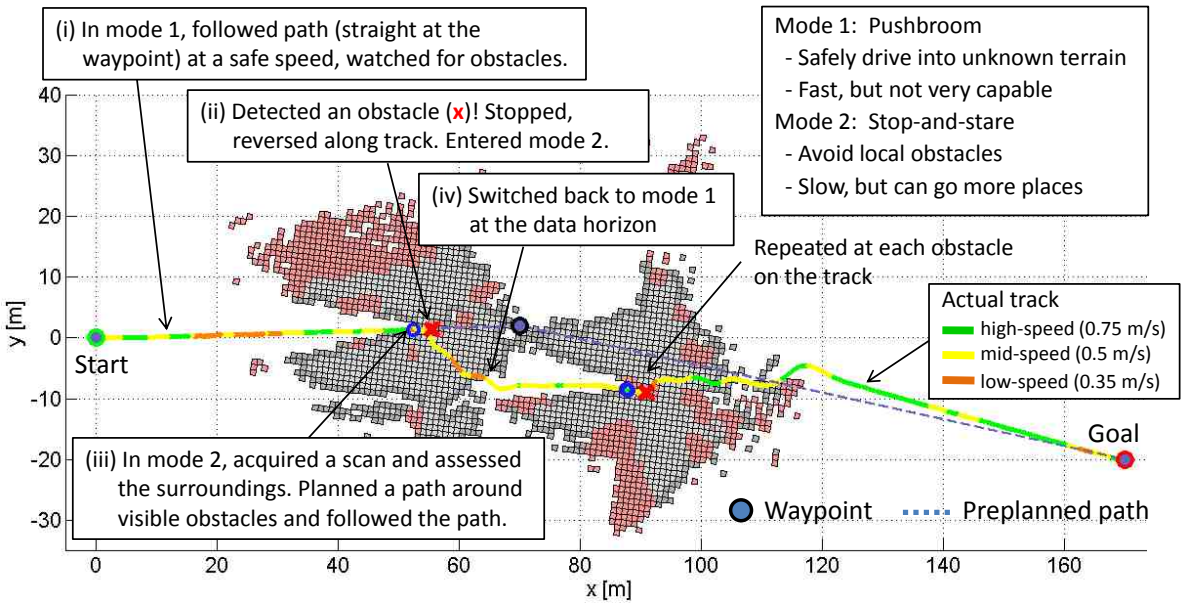


**Figure 3.1:** *A typical scenario showing the high-level operations of the short-range guidance system that was onboard the test robot. The robot was given a long-range plan consisting of a set of waypoints and a goal (shown in light blue). The robot had two guidance modes and automatically switched between them. In this example the robot did the following: (i) in mode one, it drove straight to the next waypoint until, (ii) it detected an obstacle on the path. The robot automatically stopped and then reversed along its track and entered mode two. (iii) In mode two the robot acquired a detailed scan of the terrain and used a plane-fit terrain assessment technique [67] to detect obstacles. It then planned a path to the next waypoint and drove that path. (iv) When the robot reached the sensing horizon of the previous scan it automatically changed back into mode one and drove straight at the next waypoint. These steps were repeated as necessary when obstacles were detected.*

directly inspired the development of a network of reusable paths [59, 60], as presented in Part II, and more details about the lessons learned in these field experiments are given in Section 3.2. Once the vehicle had backed up, it entered the second mode. In mode two (*stop-and-stare*), the robot was stationary while acquiring a detailed (and therefore large) three-dimensional scan of the surrounding area using the SICK laser rangefinder on a pan-tilt unit. A plane-fit method, similar to Morphin [67], was used to assess the terrain and a path was planned around any visible obstacles. Once beyond the data horizon of the sensor, typically 20 m, the system automatically returned to mode one. A sample scenario is shown in Figure 3.1. The first mode was fast but could only allow the robot to go over flat, smooth terrain; the second mode was slow but could allow the robot to negotiate much more complicated areas.

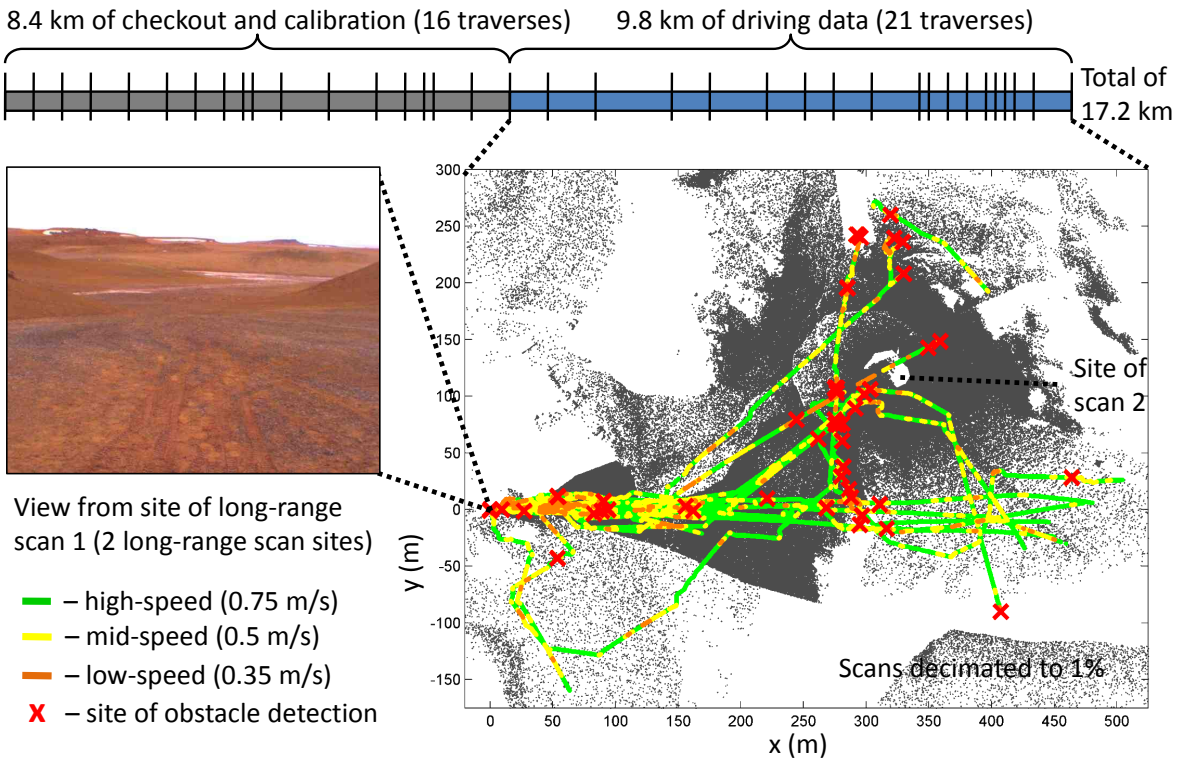


**Figure 3.2:** *Raw data collected during Devon Island field trials, July 2009. Nearly 10 km of driving data was collected over a variety of terrain. The robot's speed was recorded (tracks are colored accordingly) and so are all sites of obstacle detection while driving in mode one (shown as red 'x'). Two long-range lidar scans were taken of the traverse area.*

## 3.1 A Real-World Assessment Hierarchy

Figure 3.2 contains a summary of the performance data and long-range scans that were collected. We acquired two long-range lidar scans of the traverse area. After calibration and check-out, we collected 9.8 km of driving data over 21 traverses, and this is shown colored according to speed. A red 'x' marks a location where the robot encountered an obstacle and had to switch into mode two. We also used the information from the local stop-and-stare scans to get more data points from areas where the robot did not drive, but according to the onboard sensors, it could have, or could not have, driven.

The objective was to tie the driving data to the long-range lidar data so that we could build a classifier to predict the driving cost of moving from one cell to an adjacent cell using only the long-range lidar data. The ground-truth localization data were only used for design and validation of the system. When in operation, the SOP architecture did

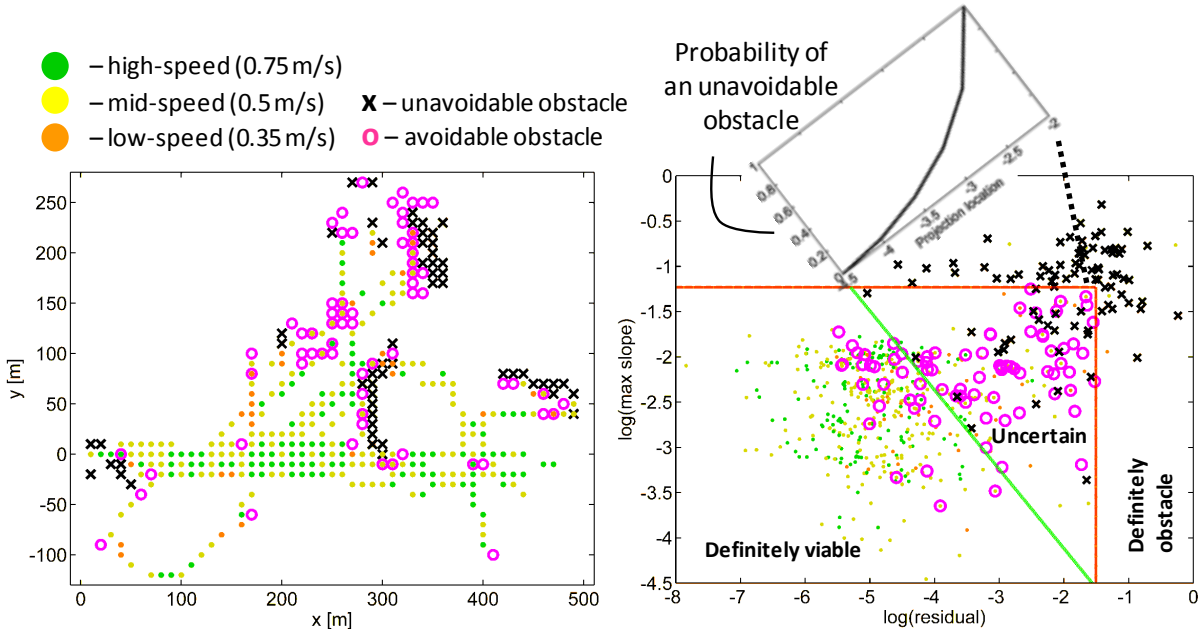


**Figure 3.3:** *The method to determine terrain costs and obstacle probabilities is shown. On the left, the mode one and two obstacle and driving data have been assigned to* $10 \times 10$ *m cells based on DGPS. Obstacles that were detected have been classified as: (i)* unavoidable obstacles *where the entire cell was not traversable (black 'x'), and (ii)* avoidable obstacles *where a cell could have been traversed by taking a path that avoided local obstacles but remained inside the cell. Other cells the robot visited are colored according to the average driving speed. Each cell also had a point cloud from the long-range lidar scan. On the right, a plane has been fit to each cell and the points from the left plot have been placed according to the logarithm of the plane's slope and roughness. A strong trend is visible and a classifier has been made creating regions where a contained cell is either viable, obstacle or uncertain. In the uncertain region the probability that a cell might be an unavoidable obstacle was modeled by projecting the fraction of unavoidable obstacles (from the ground truth driving data) onto a line.*

not need, or use, the ground-truth data. The traverse area was divided into 10 m $\times$ 10 m cells and the speed data and obstacle encounters were assigned to cells using Differential GPS (DGPS) data collected on the robot. The long-range lidar data were registered into the DGPS frame using the procedure outlined by Carle [87] to obtain ground-truth position and orientation. The points were divided into the same 10 m $\times$ 10 m cells as the driving data. This information was used to validate that the terrain-assessment methods accurately predicted terrain traversability.

In the left of Figure 3.2, the obstacle encounters have been categorized as one of two types: (i) *avoidable obstacles* that could have been avoided while staying within the cell

(a magenta 'o'), and (ii) *unavoidable obstacles* where the entire cell was unsafe (a black 'x'). Each cell was scanned at least once with the long-range lidar, and these data were used in a plane-fit terrain assessment, where a single plane was fit to all the data in the cell, and the log of the maximum slope and residual were recorded. A strong trend emerged when plotting the speed and obstacle encounters using the associated slope and residual (right of Figure 3.3), and a simple classifier was created to filter cells into viable, uncertain and obstacle categories based on the slope and residual of the plane. Within the uncertain region, the probability that a particular slope/roughness pair might be an unavoidable obstacle was modeled and predicted by looking at the distribution of the population when projected onto a line that gave a high between-class variance (top right of Figure 3.3). Online calculation of the probability was inexpensive: the operations consisted of (i) the projection onto the line, and (ii) a table look-up to find the probability given the projection distance along the line.

Unsupervised machine learning techniques may also be useful for modeling the obstacle probability. In that case, the system could use the driving-time safety checks (e.g., roll, pitch, slip detection, or the push-broom check in this system) as feedback to the online learning system. The data for the classifier presented in this section were collected in a similar way, though the classifier was created offline. Since the robot was able to detect and stop before these immediate hazards, the system was able to make assessment mistakes safely. We commanded the robot to go into questionable terrain so that we could collect the traversability data.

The high-fidelity method must be able to detect the unavoidable obstacles in the uncertain region by determining whether it is possible to move from an adjacent cell into an uncertain cell. For this system, we used the same terrain-assessment and path-planning system that made up mode two (stop-and-stare) of the guidance system onboard the robot. The high-fidelity assessment method used the point clouds from the head and tail of an edge with a much finer resolution plane-fit assessment method (0.91 m cells) to plan a path. The system could have used entirely different assessment paradigms, had they been available. Figure 3.4 is a graphical view of the terrain-assessment hierarchy used in the SOP tests. The classifier that was part of the low-fidelity terrain assessment

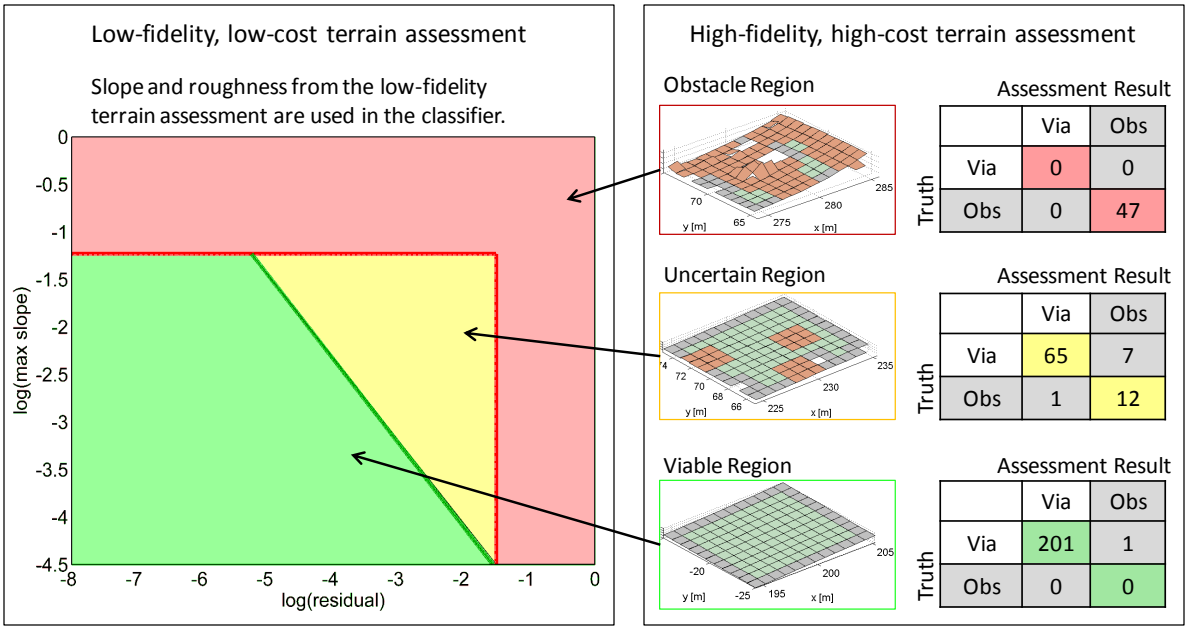


**Figure 3.4:** *A view of the real-world terrain-assessment hierarchy that was used in the SOP tests. On the left is the classifier that was used as part of the low-cost, low-fidelity terrain assessment. The terrain was classified using the position of the logarithm of the slope and roughness of a plane fit to the data. High-cost, high-fidelity terrain assessment (shown on right) yielded the same result in the viable and obstacle regions, and gave further information in the uncertain region. In the uncertain region the high-fidelity assessment method was able to identify viable terrain that could be safely traversed. The high-fidelity terrain assessment was consistent with the low-fidelity method and was quite accurate when compared to the ground truth traversability.*

is on the left. The low-fidelity method is completely described when this classifier is combined with the method of modeling the probability of an obstacle (as shown in the upper right of Figure 3.3). On the right is a visualization of the higher-resolution, high-fidelity assessments acting on example cells. For the low-fidelity assessment the log of the slope and roughness of a cell was used to classify it as viable, uncertain, or obstacle. Using the high-fidelity terrain assessment on the same cell yielded consistent results. In the obstacle region, the high-cost, high-fidelity terrain assessment found all cells to be obstacles and this was consistent with the measured data. In the viable region, almost all cells were classified as viable using the high-fidelity terrain assessment, except one, though in this case the cell was actually traversable. Similarly, there were cases for cells in the uncertain region that were actually traversable, but that were identified as obstacles by

the high-fidelity terrain assessment. This was expected, since the high-fidelity method used here did not fully exploit the capabilities of the chassis and it was conservative where it might otherwise have been uncertain. Of more concern is the case where the high-fidelity assessment classified an uncertain cell as traversable when the traverse data claims it was an obstacle. This particular error could be a result of poor data in either the traverse data or the long-range lidar scan, or it could be genuine. In any case, it is possible that terrain will be assessed as viable when, in truth, the terrain is an obstacle. This emphasizes the requirement that a robot must have a safety layer while driving (also necessary in the case of poor path tracking or localization drift). With an appropriate safety layer, the robot would still be safe, and as long as this error rate is small (and it is in this hierarchy), we assume and expect that the system will still be successful. This would be an important consideration in the design of an assessment hierarchy, but it is beyond the scope of this proof-of-concept system.

## 3.2 Important Lessons from the Field

Field tests provide an opportunity to gain unique insight into both the expected, and unexpected, interactions between vehicle and environment. These realizations can then lead to improved designs. During the course of the SOP field tests we observed behavior that led us to have a major shift in how we envision robust and capable mobile robots.

During the development of our test system, it eventually became clear that the vehicle could not reliably turn on the spot, in rough terrain, with the payloads it was carrying. We decided to allow only large-radius, sweeping turns, in order to simplify the path-tracking control system. However, this led to another problem; now the vehicle was not nimble enough to get around obstacles that it detected directly in front of itself. Since we had already eliminated the option of turning on the spot, it was clear that the vehicle was going to need to back up in order to give itself additional space to turn. The vehicle was able to drive in reverse, but it had no rear-facing sensors for obstacle detection and the vehicle blocked the terrain directly behind it from the view of the planar laser rangefinder on the PTU (i.e., the primary source of terrain data). We were trying to solve

the problem of guaranteeing safety, or at least having a reasonable expectation of safety, while the vehicle essentially drove blind in reverse. We considered the option of merging multiple scans from the stop-and-stare system and letting the vehicle drive backwards in areas the terrain assessment deemed traversable, but this also had many situations where it would not work since the stop-and-stare scans could be quite sparse and we had no expectation that the scans would overlap. Adding more sensors was not an attractive option at that late stage in the development. The only solution that had a manageable amount of work was to have the vehicle retrace its steps (i.e., back out along its path). The fact that the robot had driven over the terrain once seemed like a strong indication that it could do it again, and the platform was quite good at being able to physically drive a path both forwards and backwards.

So the system was modified to keep the last part of the track in memory where it was able to use that information to back up. The backing up was done using visual odometry, a dead-reckoning system, for localization feedback. Therefore, the localization estimate would accumulate error even as the vehicle backed up, and it would drift from the actual previous track, and the safety assumptions were no longer valid. In order to prevent this, we kept conservative limits on how far the vehicle could reverse (approximately $5-10\,\mathrm{m}$).

During the field tests, it became apparent that being able to back up arbitrarily long distances would offer great benefits. For example, the rover would be able to return home at any time, meaning that it would not get lost. Also, it would be able to escape cluttered dead-ends, even when they were too tight to allow a turn. However, this arbitrarily long repeat could not be done purely using visual odometry, because the localization error would grow the entire time and the robot would eventually drift from repeating its path.

During the same field trial another test was being conducted by a member of our lab. This was a test of *Visual Teach and Repeat* (VT&R) [2]. Visual teach and repeat is a localization and mapping system that allows a robot to repeat a previously driven route. In the experiments on Devon Island, the robot was manually driven along a long route. While it was driving, it would record stereo images. Once the route was taught, a batch operation was carried out on all the images and the result was a path with a sequence of local maps attached along it. To repeat the route, the system simply followed each local

path in sequence while localizing against the appropriate local map. The result was that the system could reliably repeat arbitrarily long paths that had been previously driven.

Seeing this visual-teach-and-repeat capability, particularly when contrasted against the challenges that a mobile robot experiences, motivated the work in Part II of this dissertation. We recognized that by extending the simple chain of maps in visual teach and repeat to work on an arbitrary network of local maps, and by allowing the robot to teach itself new paths as it was autonomously driving into new terrain, we could create a robust new navigation strategy that overcame many of the challenges with autonomous goal-seeking. The result was *a network of reusable paths*, and this is discussed next.

# Part II

# Localization Method as an Action

# Chapter 4

# The Development of a Network of Reusable Paths

We are concerned with navigating mobile robots in large-scale, unstructured, three-dimensional environments, without prior maps, and using only onboard sensors. This scenario is widely applicable indoors, underground, underwater, in urban canyons, in forests, and on other planets. There are many good techniques for navigating a robot when both the robot's position and a map of all the obstacles are known. In unmapped environments and with no global positioning, the situation becomes extremely challenging and, in a sense, is the fundamental problem of mobile robotics. *Exploration* is comprised of three issues: *localization*, *mapping*, and *planning* (see Figure 4.2). Creation of a unifying exploration framework, that is efficient enough to deal with all three of these issues simultaneously and online for large-scale, unstructured, three-dimensional environments, is an open problem in mobile robotics. The first steps toward such a framework are precisely what are presented here. This chapter presents previous work on the development and testing of the concept of a *Network of Reusable Paths* (NRP) [59, 60]. Figure 4.1 shows a robot operating on a NRP during field tests. In Chapter 5, we present the extensive hardware experiments that make use of two real visual-teach-and-repeat implementations[1]. Chapter 6 discusses how NRP can be used in planetary exploration [62, 63].

[1] A video of a robot using a network of reusable paths to reach a distant waypoint is available at: http://youtu.be/0NmpSBA1XQM. More related videos are available at: www.youtube.com/utiasASRL

**Figure 4.1:** *A robot operating on a network of reusable paths in Sudbury, Ontario, Canada.*

In this chapter, we take the position that the most important measure of success when attempting to reach a desired location is not the path length or time taken, but instead how close the robot is to that location at the end of the traverse. This is different from most path-planning paradigms (including the one in Part I of this dissertation). Some path planners look for an optimal path (according to the map representation) [50, 77, 52], and others, a path that is good enough given the available planning resources [51, 88, 89]. However, planning for minimal distance or time, and planning for minimum uncertainty can be conflicting efforts [90].

## 4.1 Introduction

To build a complete exploration system, we must consider the three main issues noted previously. The following discussion first considers the available localization methods,

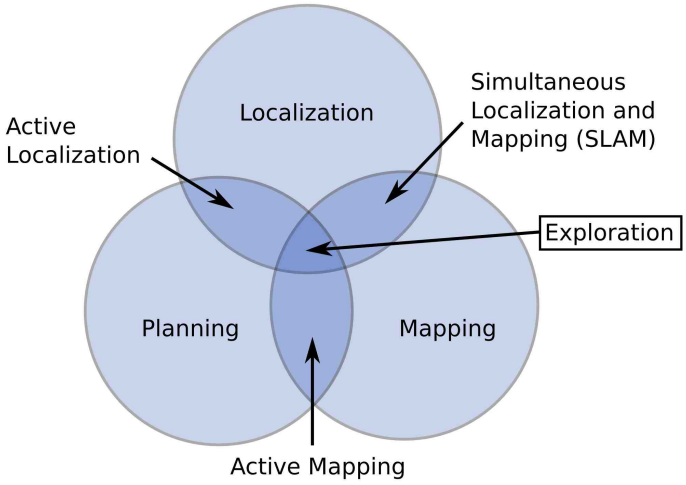


**Figure 4.2:** *Exploration is comprised of three main issues: localization, mapping and planning. Adapted from Makarenko, et al. [1]*

which leads to the desire to use a purely relative approach to map-building for localization, terrain traversability, and setting waypoints (target positions, goals). We later use these ideas to build a system that does planning for real robots in unstructured terrain.

The primary motivator for this work has been mobile robots for planetary exploration. However, the approach we use may have a more general application to robots without access to real-time global localization and with only local (myopic) sensing. Global localization, such as from the Global Positioning System (GPS), orbital observation, or fixed infrastructure, is not always available. In particular, GPS is unavailable indoors, underwater (beyond a few meters), underground, under dense foliage (a tree canopy), and on other planets. It also fails to work in areas of significant topological relief (e.g., steep terrain, urban canyons) due to multi-path effects. Myopic sensors are common onboard robots (e.g., cameras, lidar, sonar, radar, bumpers, etc.), and without global localization they must be relied upon for localization in addition to terrain assessment. Typically, these systems do not have access to a complete and accurate map of the environment a priori, thus they must build and maintain a map, as well as report their localization relative to the map.

Thus, one can consider many potential areas beyond planetary exploration where this work may be useful, some of these include:

1. *A search and rescue or disaster response scenario where the environment is unknown a priori and the robot may operate in buildings and rubble, which makes it impossible to rely on GPS.*
2. *A forestry scenario where the robot operates under dense tree-cover.*
3. *A system for use in an underground mine where there is no fixed infrastructure (that is intended specifically to enable localization).*
4. *A underwater monitoring and repair scenario where the robot operates more than a few metres below the surface.*

### 4.1.1 Classes of Localization Methods

The Global Positioning System (GPS) is an example of absolute localization. Absolute localization methods have bounded error accumulation. However, GPS is not always available (e.g., space exploration, urban canyons, indoors), and other options such as direct observation by satellite, radio triangulation [91], or matching local terrain geometry to orbital imagery [87, 91], all take time and computational resources that make them unsuitable for real-time use.

In contrast, relative localization methods, such as visual odometry [27] and wheel odometry, typically give low-uncertainty estimates of pose transformations over short distances. However, all such dead-reckoning techniques suffer from unbounded error growth as a function of traversal distance. Unfortunately, this growth is often super-linear, mostly due to orientation errors [92], and the influence of these errors can be detrimental when attempting to reach a distant goal. The robot may think it is at the goal when it is still quite far. Improved versions of visual odometry [31], or techniques that add an absolute orientation measurement [93], may reduce the error growth to linear, but this may remain problematic over very long distances (e.g., when doing vast-scale exploration through unknown terrain), or when certain sensing modalities are not possible (e.g., no visible sun for a sun sensor, or no stars visible to the star tracker, or no magnetic field for a compass).

The primary reason exploration is difficult is that the exploring entity, be it robot or other, has only local sensing abilities. Directions of travel must be chosen before the

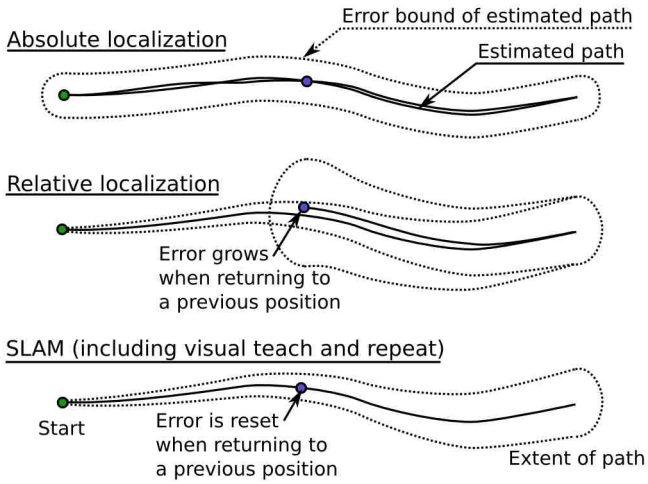


**Figure 4.3:** *Classes of localization: absolute, relative, and Simultaneous Localization And Mapping (SLAM). SLAM includes visual teach and repeat. The robot started on the left, moved to the right, and then backed up along the path to a point in the middle. Absolute localization (e.g., GPS) has bounded error, but this type of localization is not always available. Relative localization incurs error growth on the outbound and return legs. Simultaneous Localization and Mapping (SLAM), including visual teach and repeat, a type of SLAM, has only error growth when entering new terrain. When repeating a previous path the error is rolled back.*

whole environment has been observed. This leads to dead ends, forcing retreat to an earlier position. With only relative localization, the pose error will grow during retreat, in addition to the outbound trip, as in the middle of Figure 4.3.

There are, however, techniques that lie somewhere between the absolute and relative localization classes. *Simultaneous Localization And Mapping* (SLAM) systems build and maintain a map of the environment and report localization relative to that same map. These fundamental tasks were identified by Chatila and Laumond [94] as early as 1985. Shortly after, Smith, Self, and Cheeseman [95] presented a *stochastic map* representation to capture explicitly the uncertainty in the spatial relationships of interest that were contained in the map. In that system, the locations of objects in the map were reported with respect to the world reference frame (a single privileged coordinate frame). Much of the early progress also used a single privileged coordinate frame, and a thorough review of these works is done by Durrant-Whyte and Bailey [32, 33].

There have been many improvements upon single-coordinate-frame SLAM. Most of these have been attempts to reduce the large computational cost that is experienced when the map is scaled to the size necessary for a robot traveling reasonably long dis-

tances. These approaches include using *submaps* [40, 96, 97], *hybrid metric-topological SLAM (atlases)* [37, 98, 99], *hierarchical SLAM* [38], *manifold maps* [100], and *skeleton graphs* [101], amongst others.

Howard [100] introduced the idea of embedding a robot path in a higher-dimensional *manifold map* such that a single physical location could be represented by multiple points in the map. This avoids the need to make the map consistent in a single privileged coordinate frame and, importantly, retains the path driven by the robot. Sibley et al. [41] take a relative approach to bundle adjustment that uses a *pose graph* built of relative transformations between poses (and landmarks). This greatly improves the efficiency of the SLAM system as it emphasizes making the map locally, not globally, consistent. It allows for arbitrarily large maps, includes loop closure, and also retains the path of the robot. Sibley et al. [41] have also suggested the idea of planning on a relative map once it has been constructed, but do not carry out planning during the SLAM process.

A common element of the successful approaches to scaling up the SLAM problem has been allowing for the use of multiple local reference frames. It is interesting to see that there have long been arguments against trying to build a globally consistent map (for localization or traversability) in some privileged coordinate frame. Brooks [102] suggested a relative approach as early as the 1980s. These ideas extend beyond representing the pose of the vehicle or the landmarks used for localization. A relative approach is also useful for expressing local traversability maps and distant waypoints.

### 4.1.2 Visual Teach and Repeat

*Visual-teach-and-repeat* (VT&R) systems [2, 103], can also be thought of as carrying out SLAM, though perhaps not in a way that is immediately recognized. These systems allow robots to drive arbitrarily long distances without the use of GPS along previously established routes (typically the path is first driven manually in order to teach the route). In these systems, a chain of small maps is attached along the robot's path (estimated using visual odometry) during a teaching phase (i.e., it is simultaneously mapping and localizing); to repeat the route, the robot simply localizes against each small map in sequence as it drives (i.e., localization). In this way, the pose estimate will accumulate

error only during the teach pass. At any time the robot can return to a previous point on the path, and the pose error will essentially return to what it was when that section was taught. This is shown at the bottom of Figure 4.3. Visual teach and repeat remembers the path that was traveled, and the case for doing so is strong; *knowing that the robot has already successfully driven the path is strong evidence that the path is traversable.*

There are other approaches to teach and repeat. One makes use of a planar laser rangefinder for underground mining applications [20]. Another example uses an omni-directional camera [104], and still others use different techniques that provide a similar teach-and-repeat function [105, 106, 107, 108, 109, 110].

The capability of the system of Furgale and Barfoot [2] has been demonstrated through 32 km of autonomous driving in both an urban setting and a planetary analogue environment in the Canadian High Arctic. During these tests, the paths were taught manually by the operator using a handheld controller while walking near the rover, the paths were then autonomously repeated. The system, making use of a stereo camera, often performed so well that it repeated the path in its own tracks. It autonomously drove all but a few tens of metres of the desired 32 km of paths ($< 0.4\,\%$ of distance traveled).

A key to this high rate of autonomy is that the robot was frequently able to localize against the taught map. Figure 4.4 is reproduced from Furgale and Barfoot [2], and it shows the ability of the system to localize against the taught map when the pose of the camera during repeat was subject to deviations, both lateral and in orientation, from the pose of the camera in the teach pass. The local maps consisted of a cloud of SURF features that were extracted from the stereo camera images during the teach pass (an example from the current system, shown in Figure 4.5, is discussed later in this section). In Figure 4.4, the mean inlier feature count (a higher count means lower localization uncertainty) drops as the pose of the camera deviates from the nominal pose. This shows that in order to localize successfully against the taught map, the robot must closely return the sensor to the pose from which the map was taught. This makes sense intuitively, as the appearance of the scene changes when perceived from different places. Thus, in order to localize successfully and frequently along the sequence of local maps, the robot should

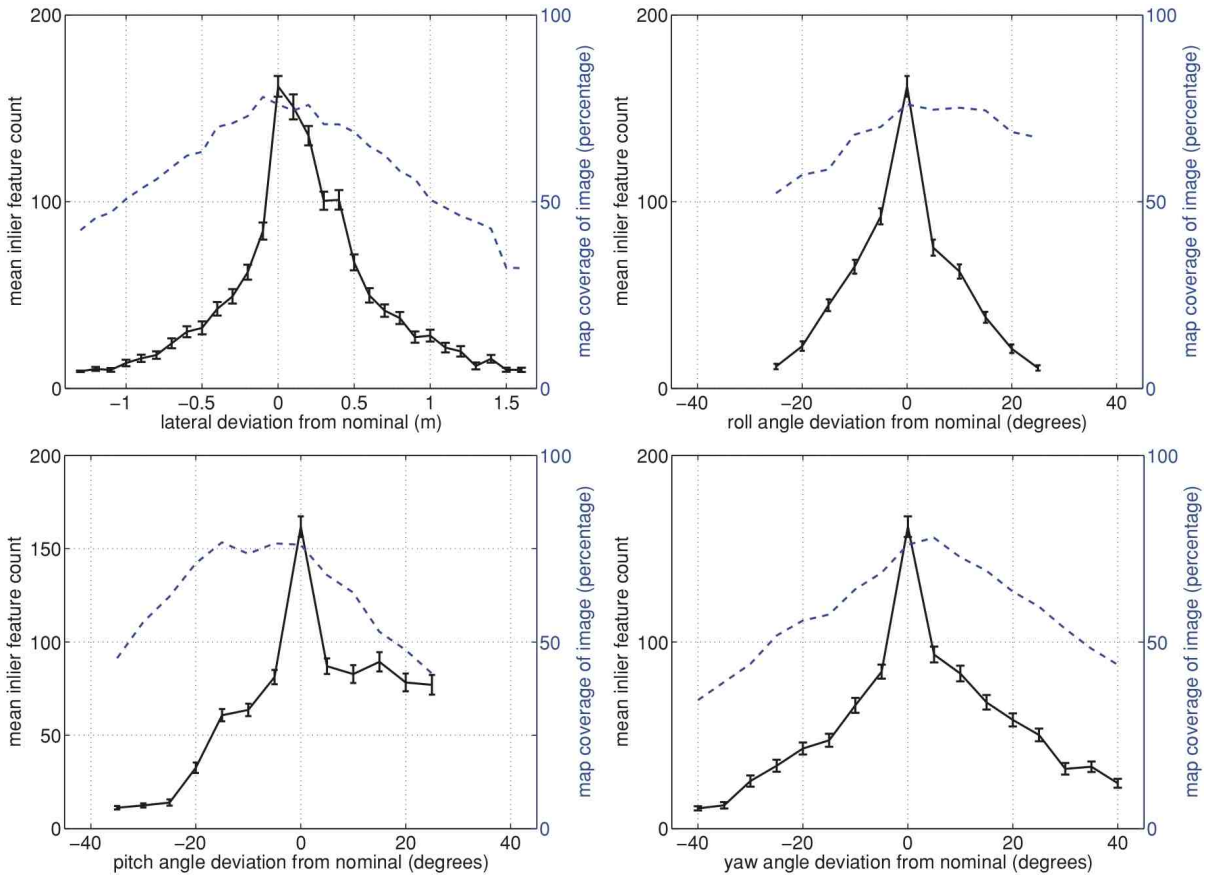


**Figure 4.4:** *An investigation into the ability of stereo-camera-based VT&R to localize against the taught map (by matching currently observed features to the taught features) subject to translation and orientation deviations from the taught pose. Figure reproduced from Furgale and Barfoot [2]*

return, as closely as possible, along the original path. *The localization against the map is more reliable when the robot travels along the original path.*

Many SLAM systems rely on large-scale loop closure, where the robot is tasked to recognize the same scene from two different vantage points, in order to do batch pose refinements. VT&R instead attempts to solve only the more simple case of small-scale loop closure, where the robot must recognize the same scene from nearly the same pose. This small-scale loop closure has a low computational cost and a high rate of success.

A lighting-invariant extension to this work, using a high-framerate lidar [103], has been developed to address one of the major challenges of the stereo-camera-based system [111], namely, appearance changes due to changing lighting conditions can make it challenging, or impossible, to localize against the map when revisiting places. This

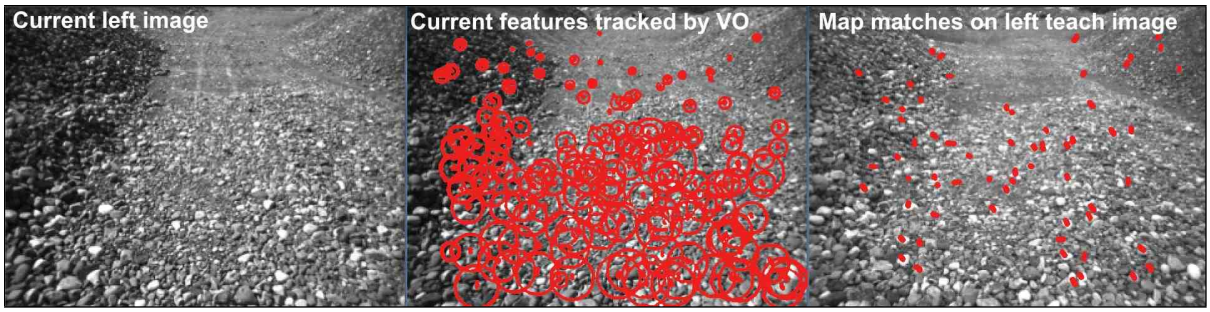


**Figure 4.5:** *Sample images from the visual-teach-and-repeat system onboard the test robot. Visual teach and repeat allows a robot to repeat arbitrarily long, previously driven routes. On the left is an image from the current position, and in the middle is the same image overlaid with the currently tracked visual landmarks (SURF features). The right image is from the initial (teach) pass when the robot first visited that position and added it to the network of reusable paths. The pairs of points in the right image show the matches between the current and taught visual landmarks. These matches are used to estimate the current pose relative to the local map. All of these images come from the left camera of the stereo camera pair.*

system allows for operation in complete darkness, making it suitable for exploration of permanently shadowed regions such as those at the Lunar South Pole.

Two real-world visual-teach-and-repeat systems are used in this work. Both of these systems are based on the work of Furgale and Barfoot [2]. The first system uses a stereo camera (this is essentially the second version of the one created by Furgale and Barfoot [2]), and the second system uses a high-framerate scanning lidar [103]. Three images from the stereo-camera-based system are shown in Figure 4.5. In the figure, the robot is in the process of repeating a path. The left image shows the current image that was just captured by the camera. The center shows the same image with the currently visible features (SURF features) superimposed. The right image is the one that was used when teaching the route (i.e., the image from which the map of the visual landmarks was derived). It shows the correspondences between the visual landmarks in the map and the features that are visible in the current frame. All images are from the left camera of the stereo pair. The geometry and the uncertainty of these point matches are used to calculate the current pose relative to the map. As in the original version of VT&R, random sample consensus (RANSAC) [112] is used to reject outlying point matches, and if insufficient matches are found, the robot will simply rely on visual odometry for short distances (i.e., it will follow the estimated path using dead-reckoning) until the currently

visible features again substantially match the features in the map. These techniques increase the robustness to changes in the scene and minor deviations from the path.

In these systems, the path is stored as a chain of vehicle poses connected by relative pose transformations determined by visual odometry. Attached to each pose is a local map of the visual landmarks observed at that vehicle pose. To repeat a path, the robot drives along the path defined by the poses and localizes against each local landmark-map in sequence. When a robot uses VT&R to reverse along a previous path, the localization error and uncertainty are rolled back to what they were when the route was taught. Other single-coordinate-frame SLAM systems also inherently allow the localization uncertainty to roll back (as expressed in the base SLAM frame) as the robot returns to previously observed areas. However, many of these SLAM systems silently forget the actual track of the vehicle, and the track is very useful (SLAM systems that use a pose graph *do* store the path). In contrast to the other approaches to SLAM, the VT&R approach has only a small computational overhead in addition to that of visual odometry. This is largely due to the use of relative coordinates and the fact that we make no attempt at loop closure as our map does not ever need to make sense in a global frame (although our map substrate is compatible with incorporating large-scale loop closure, if desired). This low overhead is beneficial since many robots, particularly those rovers used for planetary exploration, lack the computational resources to carry out other types of SLAM.

A major contribution of this chapter comes from recognizing that we can extend visual teach and repeat, from using a simple chain of local maps, to an arbitrary network of local maps. The result is a network of reusable paths [59, 60], and there are three major benefits that come from reusing paths:

1. *Traversability: Since the path has been driven, it is likely that it can be driven again.*
2. *Localizability: Since the robot is returning to a previous pose, it is very likely that the appearance of the scene will be similar and the robot will be able to localize against the map.*
3. *Rolling-back localization error: Since the localization error is rolled back when reversing along the path, dead ends do not have any influence on the accumulated localization error at the goal.*

We believe that exploration is still an open problem in mobile robotics; there are few examples of tightly coupled localization, mapping, *and* planning, for large-scale, unstructured, three-dimensional environments, particularly without using GPS. The challenge remains to find an efficient autonomous exploration technique appropriate to online, real-world use. This part does not address all of these aspects, but it does lay the foundation for future works.

### 4.1.3 Seeking a Goal in Unknown Terrain

Consider the classic scenario of a robot seeking a distant goal (or waypoint) location (defined relative to its start location) through unknown terrain. To do this well in a single reference frame, in large-scale, unstructured, three-dimensional environments, using existing robotic tools, we could try to combine (i) a sophisticated (single-coordinate frame) SLAM system to build a landmark map and localize (as we drive into cul-de-sacs and retreat), (ii) a sophisticated terrain-assessment system to build an globally consistent obstacle map from the SLAM localization, and (iii) a sophisticated path-planning system to generate smooth collision-free kinematically feasible trajectories on the obstacle map. The problem with this paradigm, is that all of these techniques are computationally very expensive, yet they are *all* necessary to create a working system.

By contrast, we show that by extending visual teach and repeat to systems that use a network of reusable paths, and coupling NRP with a path planner that can choose to either (i) drive on the known network of reusable paths (including in reverse to backtrack out of cul-de-sacs), or (ii) drive into unknown terrain (and therefore add to the known network of reusable paths), results in a navigation framework with the following desirable properties:

1. *Sophisticated localization/mapping (i.e., detecting and closing loops) is helpful, but only simple visual odometry and the ability to remember landmarks (relative to a path) is necessary.*

2. *Sophisticated terrain assessment for obstacle detection/mapping is helpful, but only simple terrain sensing (e.g., a bumper) is necessary.*

3. *Sophisticated path planning to generate smooth collision-free kinematically feasible trajectories is helpful, but only the simple ability to randomly branch from the known NRP is necessary.*

We believe this approach (continuing the trend away from building maps in a single privileged coordinate frame, and adding the concept of growing a network of reusable paths) enables *computationally efficient*, *accurate*, and *robust* navigation of mobile robots in real-world environments.

The remainder of this chapter is as follows: Section 4.2 presents some background. In Section 4.3 we develop how to use and plan on a network of reusable paths. Simulations are presented in Section 4.4. We detail two sets of field tests in Chapter 5, which also includes a discussion on the performance of NRP.

## 4.2 Review of Planning for Reduced Localization Uncertainty

This review addresses three approaches to planning under localization uncertainty. These methods attempt to improve the performance metric with which we are concerned; that is, they attempt to reduce the distance the robot is from the true goal at the end of the traverse.

The first approach is to consider localizability during path planning. One method is to plan a path and then check that there are sufficient landmarks along that path for the robot to localize [113]. Coastal planning [114] brings the test for localizability into the planning step so that landmark visibility is considered during planning. However, this imposes a significant preprocessing cost for any potential terrain that may be considered [115].

The second approach is to consider where the robot may actually drive by trying to anticipate possible localization drift. Particle-based Rapidly-exploring Random Trees (RRT) [116] and robust planning [117] both enable path planning that accounts for accumulation of error in the pose estimate as the robot drives a path. This helps avoid collisions even when the robot is unable to track the desired path because the relative

pose estimate has deteriorated.

The third, and final approach mentioned here, is planning in belief space [115, 118]. Gonzalez et al. [115] present a resolution-optimal path planner that models pose uncertainty. The method puts an uncertainty constraint on the goal and then looks for a low-cost (in time, distance) path to the goal. In effect, the footprint of the robot increases with distance along the planned path, therefore, the planner prefers open areas when the pose uncertainty is large. This reduces the chance that the robot will encounter a known obstacle, even when the localization estimate drifts from the true pose. The approach allows for intermittent use of GPS, so a path may divert to an area with GPS in order to reset the localization error to near zero. The environment is assumed to be known. More recently, belief road maps [119] offer a more efficient way of planning in belief space after incurring an initial preprocessing cost.

The problem is that these solutions do not scale computationally, or, due to the large upfront costs, they are better suited to cases in which the robot plans many times on the same map, rather than exploration problems where a robot is regularly entering previously unvisited areas. We take an approach that avoids much of the computational burden by using a navigation strategy that makes use of a network of reusable paths. *This allows a robot to plan based on how it has done, rather than considering all the possible outcomes of how it may do.*

## 4.3 Using a Network of Reusable Paths

This section presents our main contribution, the network of reusable paths. First, we lay out the assumptions we make about the robot and the environment (Section 4.3.1). Then, we introduce the concept in a sample scenario (Section 4.3.2) in order to provide some intuition into the paradigm. Next, we give the high-level algorithm (Section 4.3.3) in order to set up the definition of the network structure (Section 4.3.4) and the path planning algorithm (Section 4.3.5). Terrain assessment is incorporated in Section 4.3.6.

### 4.3.1 Assumptions About the Mobile Robot

Two assumptions are made about the robot:

1. *The robot is able to detect and stop before hazards,* e.g., bumper, rollover detection. More sophisticated terrain assessment can also be incorporated, if available.
2. *The robot can always travel along a previous route, in both directions, but with the same orientation.* This is enabled by using a visual-teach-and-repeat system (and therefore implicitly assuming the local appearance of the scene does not change, this in turn leads to the assumption that the geometry of the scene is static). Recovery in the event that this is not possible is an important consideration. However, with the visual-teach-and-repeat system considered, this failure is rare ($< 0.4\,\%$ of distance traveled, and if the requirement is that the robot be able to travel in at least one direction, the failure rate is even lower [2]) so the simplification is justified at this stage. There are typically two ways that this assumption is broken [111]:
   (a) The appearance of the scene changes such that the system cannot localize against the map.
   (b) The path cannot be reversed (for example after sliding down a steep hill or going down a high step), or the traversability of the path changes so that it is no longer traversable, (e.g., the path becomes blocked).

### 4.3.2 Sample Scenario

Returning to the scenario of seeking a goal in unknown terrain, a basic version of our proposed network-of-reusable-paths framework could combine (i) visual teach and repeat (on a network of reusable paths) for localization/mapping, (ii) a very simple terrain-assessment method (e.g., a crude safety monitor to check for an obstacle directly ahead or danger of rollover using an inclinometer), and (iii) a very simple path planner (e.g., go to a random location on the known network of reusable paths and then drive a random path into unknown terrain). Because NRP allows the robot to return along all or part of its outbound path when dead ends are encountered, it will never become hopelessly lost and it will only incur growth of localization error along the best route it finds to

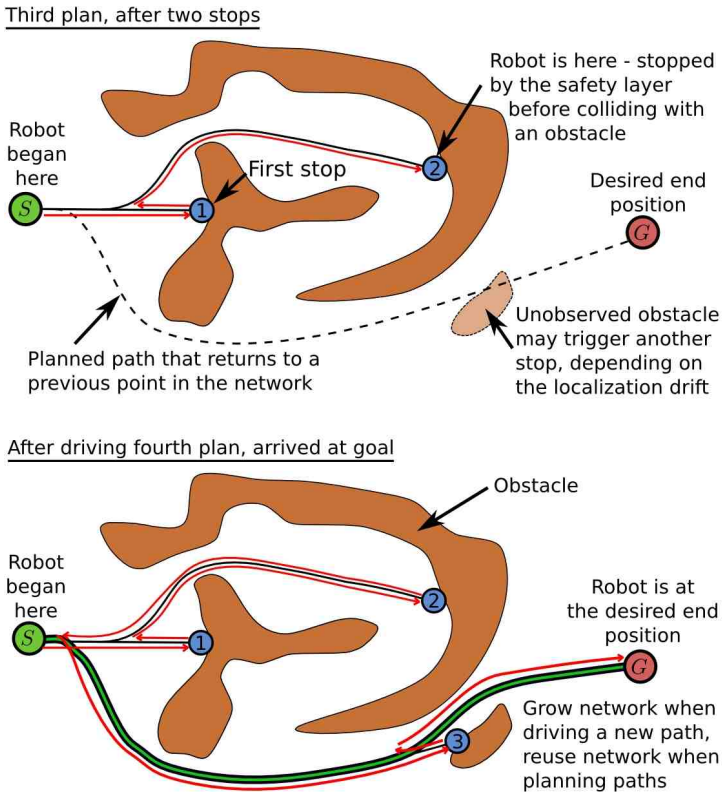


**Figure 4.6:** *A simple scenario using the network-of-reusable-paths paradigm to navigate from the start, $S$, to a desired end position or goal, $G$. In the top, the robot has a plan that uses the network of reusable paths to backtrack before breaking off into previously untraveled territory. In the bottom the robot has made it to the goal. The red arrows show the sequence of where the robot drove. Localization error at the goal is only accumulated along the green path.*

the goal. Swapping in more sophisticated component technologies is compatible with the framework, and it would undoubtedly help the robot find the goal more quickly; however, the critical point is that high accuracy and robustness are achievable using relatively simple and computationally efficient techniques within the NRP framework.

Let us consider the sample scenario in Figure 4.6. The robot used the NRP paradigm to navigate into unknown terrain. The robot began at the start, $S$, and a desired position (a goal), $G$, was defined relative to this initial pose. The robot began with no knowledge of the terrain and all obstacles were out of sensing range, so the robot simply drove a straight line path toward the goal. It stopped when it detected an obstacle. It then backed up, and went around to its left until it encountered another obstacle. This is the scene at the top of Figure 4.6. The planner was then able to reuse the previous network of reusable paths to backtrack, and roll back the localization error, before attempting the

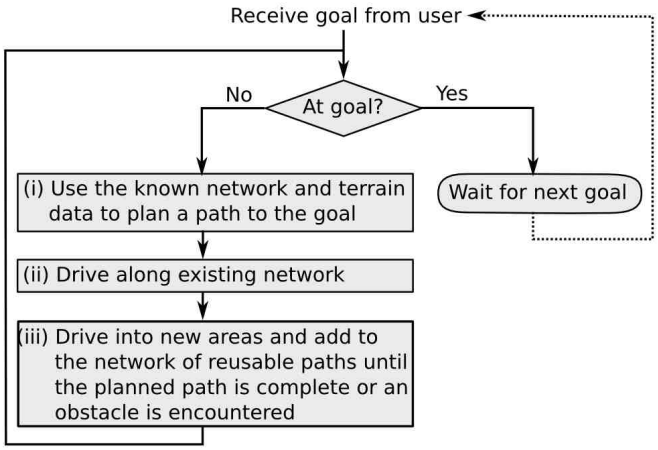


**Figure 4.7:** *High-level algorithm for operating on a network of reusable paths.*

new plan around to the right. Before the robot reached the goal, it detected one more obstacle and created another new plan. In this example, the robot has only accumulated localization error along the green path from the start to the goal, despite having driving much farther.

### 4.3.3 The High-Level Algorithm

Consider a robot tasked with reaching a distant goal through unknown terrain. The network of reusable paths paradigm is used in the construction of a high-level algorithm for seeking that distant goal. The algorithm, as shown in Figure 4.7, has three steps that are repeated until the robot reaches the goal: (i) plan a path using the known network and the available terrain data, (ii) according to the plan, drive along the existing network, and (iii) according to the plan, drive into new areas and add to the network of reusable paths until the path is complete or an obstacle is encountered. In order to complete these steps we must have a definition of the mathematical structure underlying a network of reusable paths, so that the robot can localize on the network, add to the network, attach local terrain assessment data, and plan paths along and off the network.

### 4.3.4 The Network Structure

The NRP is represented by a graph, $\mathcal{G}$, that consists of a set of nodes, $V$, and a set of edges, $E$. This is shown in Figure 4.8. Each node represents a previous pose, and the

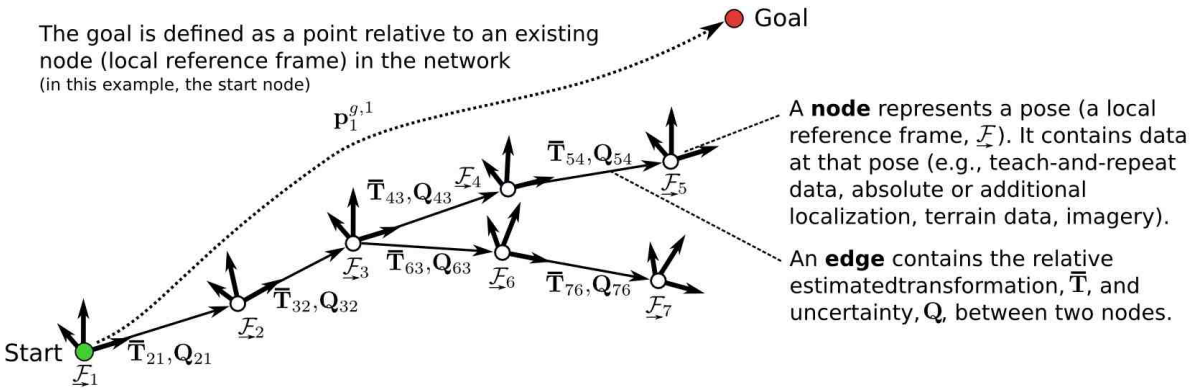


**Figure 4.8:** *Definition of the key components of the graph structure used in a network of reusable paths. There is no privileged coordinate frame, everything is relative.*

node contains the local visual-landmark-map (used by VT&R) associated with that pose. All poses are relative, with the estimated mean transformation, $\overline{\mathbf{T}}$, and the associated covariance matrix representing uncertainty, $\mathbf{Q}$, stored at an edge connecting two nodes. The actual transformation, $\mathbf{T}$, is unknown. The relative transformation and uncertainty between any two connected nodes can be found by compounding the relative transforms (and uncertainties) along a chain joining the two. The result is a graph similar to the paradigm of Sibley et al. [41], in that there is no privileged coordinate frame; everything is relative.

Taken together, the nodes and edges are the paths the robot has previously taken; based on existing VT&R capabilities, we assume that these paths (and subsets) can be repeated exactly, in either direction, with the robot in the same orientation as the initial (teach, mapping) pass. Additional information can be stored at the edges or the nodes (e.g., terrain assessment data, absolute localization, localization in other frames, sensor data). The goal, $G$, is designated as a point in one of the reference frames at a node, called the goal definition node, $x_{\text{gd}}$. This is not necessarily the initial robot pose, $x_0$, or the current robot pose, $x_r$.

There is also a cost for each of the edges in the graph. With this cost in place, we can construct a minimum spanning tree of the connected graph by using a graph-search technique [49, 120]. The minimum spanning tree is later used during planning. Note that the test systems presented in these chapters are restricted to a network that is a tree (i.e., no loop closure); however, the concepts that we describe apply equally to arbitrarily

connected networks

With this structure in place, we can construct the following rather simple theorem about the localization error. Though somewhat obvious, it nonetheless is fundamental to the performance of NRP and captures the basic benefit of using a purely relative approach to exploration. A visual showing the implications of the theorem and the associated remarks is shown in Figure 4.9.

**Theorem 4.1.** *The localization error at node $x_b$ is only due to contributions along the chain of poses that connects $x_b$ to the node from which the localization is expressed, $x_a$.*

*Proof.* A minimum spanning tree can be constructed from the network. With this tree structure, there is exactly one cycle-free path between any two nodes. The path from node, $a$, to a different node, $b$, is a sequence of $N$ nodes

$$P_{b,a} = (x_{p_1}, \ldots, x_{p_N}) \tag{4.1}$$

where $x_a = x_{p_1}$ and $x_b = x_{p_N}$. The transformation to the node $x_{p_N}$ from the node $x_{p_1}$, is then composed of all the transformations, in sequence, on the path so

$$\mathbf{T}_{p_N,p_1} = \mathbf{T}_{p_N,p_{N-1}} \ldots \mathbf{T}_{p_2,p_1}. \tag{4.2}$$

Similarly, the estimated transformation is

$$\overline{\mathbf{T}}_{p_N,p_1} = \overline{\mathbf{T}}_{p_N,p_{N-1}} \ldots \overline{\mathbf{T}}_{p_2,p_1}. \tag{4.3}$$

The error transformation is

$$\widetilde{\mathbf{T}}_{p_N,p_1} = \overline{\mathbf{T}}_{p_N,p_1} \mathbf{T}^{-1}_{p_N,p_1}. \tag{4.4}$$

The error depends only on the transformations along the path that connects $x_a$ to $x_b$. □

This theorem leads to two related and important remarks. The first states that the final localization error depends only on the final path to the waypoint; therefore dead ends do not influence the localization error at the waypoint. The second shows that

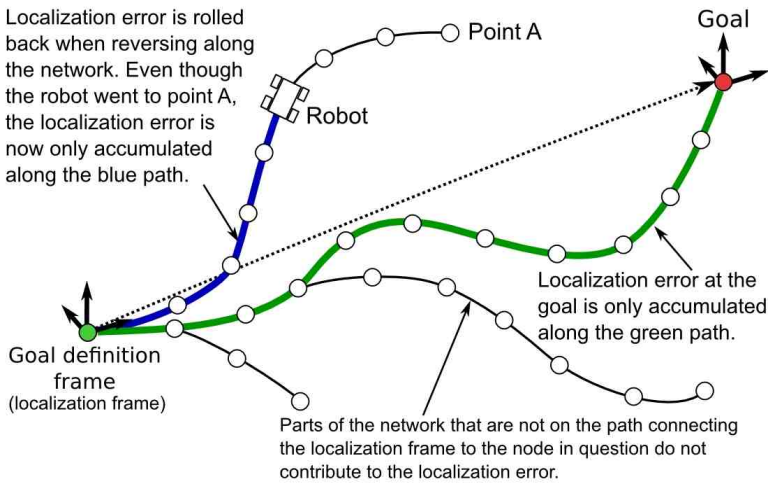


**Figure 4.9:** *Characteristics of the localization error when using a network of reusable paths. The localization error at a node is only due to the errors in the transformations at the edges that connect the node used as the localization base frame, and the node under consideration. This means the localization error at the goal is only accumulated on the final path to the goal from the goal definition node, and that localization error is rolled back when reversing along a previous route.*

localization error is rolled back when reversing along a previous path, and when the robot revisits a node, the error is reverted to what it was when that node was taught.

**Remark 4.1.** *The localization error at the node closest to the goal, $x_w$, is accumulated along the final path through the network from the goal definition node, $x_{gd}$, to $x_w$.*

**Remark 4.2.** *When localizing against a node, $x_p$, the localization error, with respect to the localization base frame at node $x_b$, reverts to what was experienced when $x_p$ was added to the network (with the addition of the error currently experienced in the localization against node $x_p$).*

### 4.3.5 Path Planning using a Network of Reusable Paths

The planning problem, given such a network of reusable paths, is to find a low-localization-uncertainty path to the goal, if such a path exists. The goal could be on or off the existing network. In this work, we assume that the uncertainty grows monotonically with distance traveled when using dead-reckoning (thus the pose error tends to grow only when new paths are added to the network), we therefore use distance as a proxy for uncertainty.

For pre-driving planning, we use a path planner that is based on a Rapidly-exploring Random Tree (RRT) [51, 61]. The original RRT algorithm is shown in Algorithm 2. The configuration space, $\mathcal{X}$, is the space of all possible vehicle configuration states (this work uses the vehicle pose). The configuration space is divided into an obstacle region, $\mathcal{X}_{\text{obs}}$, and a free region, $\mathcal{X}_{\text{free}}$, such that,

$$\mathcal{X} = \mathcal{X}_{\text{obs}} \cup \mathcal{X}_{\text{free}}. \tag{4.5}$$

In the original RRT algorithm, the tree used in planning (a graph, $\mathcal{G}$, of states, $V$, connected by edges, $E$, such that $\mathcal{G} = (V, E)$) is initialized as the set of states containing only the current state, $x_{\text{init}}$, and the empty set of edges. It then carries out the next steps for $K$ iterations:

1. First, select a random state, $x_{\text{rand}}$, from the set of configurations states, $\mathcal{X}$ (some approaches select only from the free states, $\mathcal{X}_{\text{free}}$ [53, 121]), using the RandomState() function. Note that by occasionally selecting $x_{\text{rand}} \leftarrow x_{\text{goal}}$ the growth of the tree can be biased toward the goal.

2. Second, use the Nearest() function to find the state, $x_{\text{nearest}}$, in $V$, that is nearest to $x_{\text{rand}}$.

3. Third, use the Steer() function to select a kinematically feasible edge, $e_{\text{new}}$, that extends from $x_{\text{nearest}}$ to a new state, $x_{\text{new}}$.

4. Finally, check if any part of $e_{\text{new}}$ intersects with $\mathcal{X}_{\text{obs}}$. If $e_{\text{new}}$ does not encounter an obstacle, then add $x_{\text{new}}$ and $e_{\text{new}}$ to the tree contained in $\mathcal{G}$.

The final path to the goal can be extracted from $\mathcal{G}$ by choosing the state in the tree that is closest to the goal, and then traveling back along the tree to the initial state. A robot using the RRT planning algorithm would then drive the extracted path.

There are several notable methods for improving the practical performance of RRTs in path planning. The work of Urmson and Simmons [122] biases the growth of the tree through lower-cost regions. Ferguson and Stentz [123] show an anytime-planning approach that uses many RRTs that share information to quickly create an initial plan,

**Algorithm 2** An outline of the Rapidly-exploring Random Tree (RRT) algorithm [51, 61].

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \{\emptyset\}$; $\mathcal{G} = (V, E)$
**for** $k = 1$ **to** $K$ **do**
  $x_{\text{rand}} \leftarrow \text{RandomState}(\mathcal{X})$
  $x_{\text{nearest}} \leftarrow \text{Nearest}(G, x_{\text{rand}})$
  $x_{\text{new}}, e_{\text{new}} \leftarrow \text{Steer}(x_{\text{nearest}}, x_{\text{rand}})$
  **if** $\text{ObstacleFree}(e_{\text{new}})$ **then**
    $\mathcal{G} \leftarrow \mathcal{G} \cup (x_{\text{new}}, e_{\text{new}})$
  **end if**
**end for**
**return** $\mathcal{G}$

**Algorithm 3** An outline of the RRT-based NRP planning algorithm.

$\mathcal{G}_T \leftarrow \text{MinimumSpanningTree}(\mathcal{G}, x_{\text{gd}})$
**for** $k = 1$ **to** $K$ **do**
  $x_{\text{rand}} \leftarrow \text{RandomState}(\mathcal{X})$
  $x_{\text{nearest}} \leftarrow \text{Nearest}(\mathcal{G}_T, x_{\text{rand}})$
  $x_{\text{new}}, e_{\text{new}} \leftarrow \text{Steer}(x_{\text{nearest}}, x_{\text{rand}})$
  **if** $\text{ObstacleFree}(e_{\text{new}})$ **then**
    $\mathcal{G}_T \leftarrow \mathcal{G}_T \cup (x_{\text{new}}, e_{\text{new}})$
  **end if**
**end for**
**return** $\mathcal{G}_T$

and then improve the cost of the path while time permits. Howard et al. [121] present a model-based trajectory generation approach to state-space sampling that enables better planning in cluttered or highly constrained environments. Recently, an optimal extension to RRTs, called RRT* [53], has been developed. The relationship between NRP and RRT* is explored in Section 5.4.

The steps of the algorithm for pre-driving planning, on a network of reusable paths, are shown in Figure 4.10, and alternatively in Algorithm 3. These two views show the same algorithm for planning a path. Once the plan is created, the robot drives the path according to the algorithm in Figure 4.7. The RRT algorithm shown in Algorithm 2 is slightly modified in the NRP planning approach. First, instead of initializing the tree with just the initial state (and no edges), we use a minimum spanning tree, $\mathcal{G}_T$, created from the network of reusable paths, $\mathcal{G}$, and rooted at the goal definition node, $x_{\text{gd}}$. The other difference is that instead of sampling a random state from the free configuration

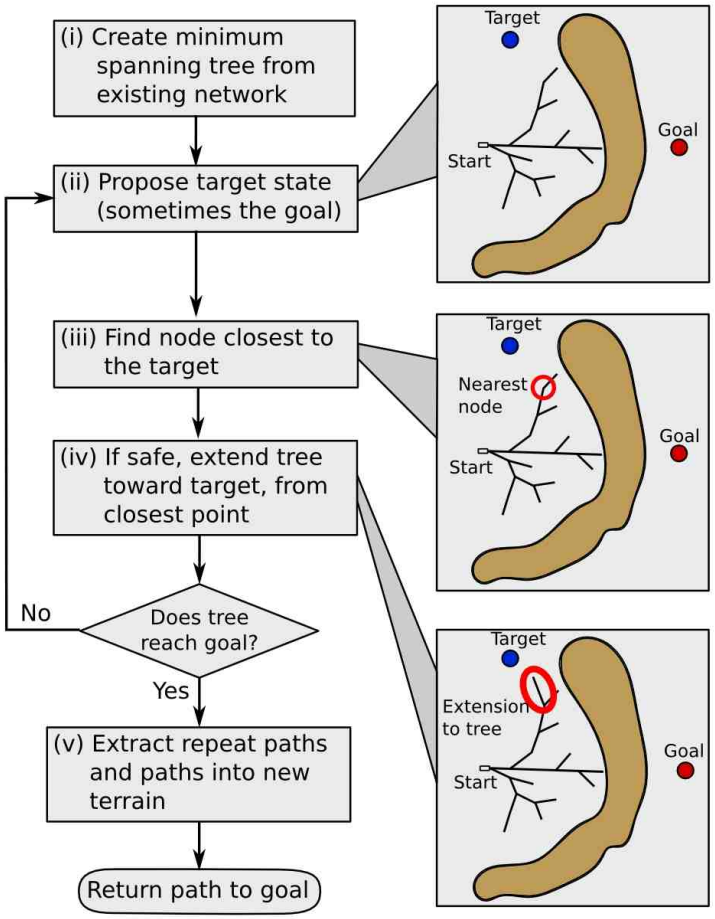


**Figure 4.10:** *An RRT-based path planner that uses a network of reusable paths.*

space, we sample a random state from the entire configuration space, because we do not have a globally consistent map of the obstacle region. We do not assume that the entire traversability map is known a priori, or that the current map is globally consistent. Therefore, we do not build a global traversability map and we cannot know $\mathcal{X}_{\text{obs}}$. We would argue that in practice no robot doing exploration truly knows $\mathcal{X}_{\text{obs}}$, but this is typically not captured by the planning algorithms in the literature. Not knowing $\mathcal{X}_{\text{obs}}$ also has implications for using the ObstacleFree() function, and this problem is addressed in Section 4.3.6.

We can now step through the algorithm according to the numbering in Figure 4.10. First, in step (i), from the network, $\mathcal{G}$, we can create a minimum spanning tree, $\mathcal{G}_T$, rooted at the goal definition node, $x_{\text{gd}}$. The metric we have used to find the spanning tree is the Euclidean distance, meant to roughly represent pose uncertainty. We assume

that $\mathcal{G}_T$ connects $x_{\text{gd}}$ to $x_r$; these may be the same node. The tree is then used as the substrate for path planning. More specifically, it is the initial tree for the RRT. In the case of a new network (i.e., there are no existing reusable paths), the initial tree consists solely of the current node.

The core RRT is contained within steps (ii) through (iv). Step (ii) is to propose a target state. Sometimes it may sample the goal; this biases the growth of the tree toward the goal. Step (iii) is to find the node on the tree that is closest to the target state.

Step (iv) is to safely extend the tree, if possible, from the closest node, toward the target state. The candidate extension is a new node, connected to the previously closest node in the network by an edge that respects the kinematic constraints of the robot (we set a limit on the minimum turn radius). Terrain assessment is used to determine the safety of a candidate extension. If the segment is safe, the node and edge are added to the tree, and these may be used in later tree extensions or as part of the plan.

If the tree does not extend to the goal (or to within some threshold distance from the goal), the algorithm returns to step (ii). Once the tree reaches the goal, the algorithm moves to step (v), which is to extract the path from the tree. The path is extracted in two parts: (i) the path that reuses the existing network (i.e., from the current point on the network to the point where the new plan departs the existing network), and (ii) the section that goes into new terrain.

### 4.3.6 Terrain Assessment

Now we return to the problem of terrain assessment (the ObstacleFree() function). In the NRP paradigm, the system uses terrain assessment data stored relative to nodes in the network. For a single path segment, there may be many nodes that contain relevant relative-assessment information. Merging multiple assessments can lead to many practical problems (discussed in Section 5.3.1), so we never merge any assessment data; we never build a monolithic map in some privileged coordinate frame. Given that we have many local maps of assessment data, instead of a global map, the system must select and check the most appropriate terrain assessment data, and safely and robustly resolve conflicts in traversability interpretations.

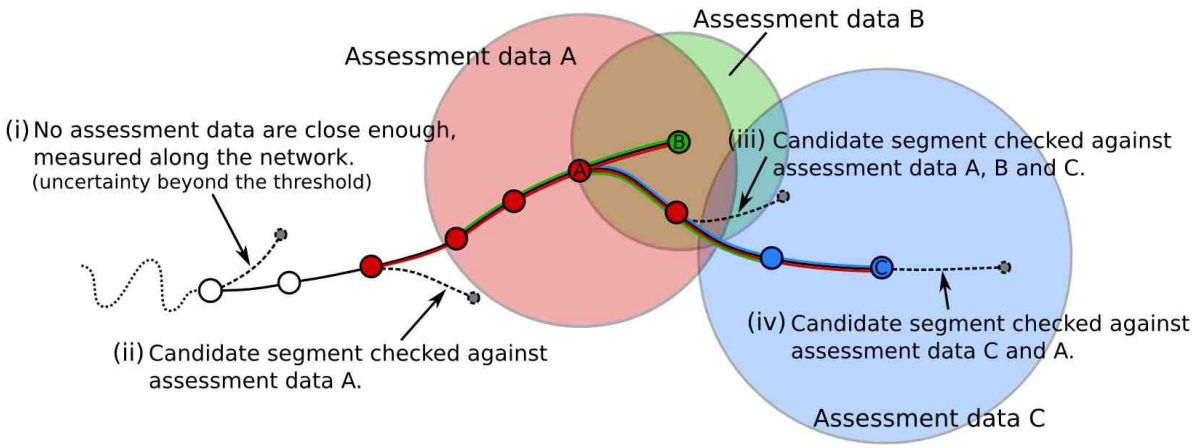


**Figure 4.11:** *A network of reusable paths with local terrain assessment data stored at the nodes. Candidate edge extensions proposed during planning are checked against topologically nearby assessment data. The order that these nearby assessments are checked is determined according to the weighted distance as calculated in equation 4.6. If no data are close enough, as measured along the network, then the candidate edge is accepted by default.*

The theme of reduced localization uncertainty applies in this domain as well. We want to use the most complete, most accurately localized assessment data, when planning the rover path. In an attempt to rank them, we give the assessment data at each node a priority score, $d_s$. The score is a linear combination of the topological distance along the network, $d_t$, and the range, $d_r$, to the segment in the assessment data frame. Thus,

$$d_s = \alpha d_t + d_r, \tag{4.6}$$

where $\alpha$ is a weighting. The assessments are then checked in the order of increasing score (i.e., lowest-score assessment data is used first). The checks are made until a threshold number of checks have sufficient data for an assessment of the proposed segment, and these checks show the terrain to be traversable. If too many of the local maps indicate the segment is not traversable, the candidate extension is discarded. Otherwise the segment is accepted and may become part of the planned path. Figure 4.11 shows four examples of candidate segments. Candidate segment (i) has no nearby assessment data, so it is assumed to be traversable. Segment (ii) is checked against the nearby assessment data A, but there are no data at the segment location, so again, this segment is assumed traversable. Segments (iii) and (iv) do have nearby assessment data that are at the

proposed segment positions. Segment (iii) will be checked against data from A, B and C, while segment (iv) is checked against C and A, in that order.

Most planning systems expect a complete and accurate traversability map of the configuration space (i.e., the robot actually has $\mathcal{X}$) by having a perfect traversability map (and possibly a perfect model of the robot). However, this is very often not available, particularly a priori. Here, the robot has an estimate, $\bar{\mathcal{X}}$, of the map, and this estimate is constructed of $N$ local, estimated submaps, $\bar{\mathcal{X}}_i$, ($i$ is the map index), where $N \geq 0$ (there may be many submaps at a single node in the network). This can be thought of as

$$\bar{\mathcal{X}} = \{\bar{\mathcal{X}}_1, \ldots \bar{\mathcal{X}}_N\}. \tag{4.7}$$

Tying the assessment data to an individual node, in the node's reference frame, makes it easy to abstract the concept of what constitutes terrain assessment. For example, one might include: availability of power, availability of communications, or ability to localize, in addition to the terrain roughness, step and slope. The planner does not require knowledge of the source of the assessment information. If additional data are needed for a certain type of assessment procedure, then those data, too, would be stored at the node and the terrain assessment method would access them as needed in order to respond to the assessment requests.

## 4.4 Simulations Comparing the Performance of NRP to that of a Benchmark Robot

We have carried out simulations of a robot navigating in fractally generated terrain. Using those simulations, we compared the performance of the robot using the network-of-reusable-paths navigation paradigm to a benchmark system made of the same basic components (minus the ability to reuse a previous path), and to a system that has prior knowledge of the entire terrain map. Here we first discuss the simulation setup, before discussing the results.

### 4.4.1 Simulation Setup

This section covers four parts: (i) how the terrain is generated and the basic robot model, (ii) the benchmark navigation system, (iii) the network-of-reusable-paths navigation system, and (iv) a system that uses a priori terrain data to plan a path.

#### Terrain Generation and Robot Model

We generated fractal terrain [83] that was modeled as a square, three-dimensional elevation map, with 513 pixels per side. The terrain sensor onboard the simulated robot was modeled after a finite-range sensor (with a 40-pixel range) such as a lidar, mounted at a fixed height above the terrain. Terrain occlusions were modeled based on the geometry of the visible terrain and the perspective of the sensor. The sensor was able to be pointed in any direction.

The terrain assessment at a pose was based on a disk-shaped robot model that used the elevation variance and maximum step height of the terrain within the robot footprint. The planner treated areas with no data, or insufficient data, in the same way that it treated areas that were found traversable according to the terrain assessment (i.e., the planner was optimistic).

The robot also had a driving-time safety layer that monitored the safety of the terrain either directly in front or behind the robot, depending on the direction of travel. This means the robot was allowed to safely travel in either the forward or reverse direction.

The turning radius of the robot was constrained. The path planner and vehicle model would not allow turns tighter than a particular radius (5 pixels). We used a unicycle model for the robot, and added zero-mean Gaussian noise to the relative translation and rotation at each time step. Effectively, this meant the pose estimate would drift from the true pose.

#### The Benchmark System

The benchmark system would plan a path and then follow that path until it encountered an obstacle. It would then acquire a scan of the terrain and merge the resulting assessment data with a window of assessments from previous scans (up to 15 scans) to build a

traversability map that the RRT-based path planner could use to find a path to the goal. The window approach allowed the rover to forget assessments that might have caused the planner to wrongly believe it was trapped. The path plan was always created from the current pose of the robot. If no path was found, the robot would forget all assessments, except for the last one, and attempt to create a new plan. If it was still unable to find a path, it reported that it was stuck. The benchmark system simply used odometry for localization, and did not have the ability to reuse old paths. This meant localization error and uncertainty were incurred along the entire path driven by the robot.

#### The Network-of-Reusable-Paths Navigation System

The NRP approach used the current network to seed the RRT-based path planner. As in the benchmark system, it would drive the planned path until an obstacle was encountered. It would then stop and acquire a scan and add the resulting local assessment map to the network. The terrain traversability was checked against the nearby scans within a certain distance threshold. The robot was always able to return along its previous paths so it was not possible for it to get stuck. The two core assumptions about the robot are listed in Section 4.3.1.

#### Planning with A Priori Terrain Data

In a third system, we allowed the same path planning algorithm that was used in the benchmark system to use all the terrain data to plan a path. It is unlikely that a real robot doing exploration would have access to all the terrain data a priori. Figure 4.12 shows the resulting paths on an example map. It is worth noting that it is not only doubtful that the robot would have access to all the terrain data; it is also doubtful that the robot would be able to follow these paths without stopping and replanning. The plans may go through small openings, and with localization drift, the robot would be unlikely to make it through on the first attempt. This a priori method provides an upper bound on the achievable performance.

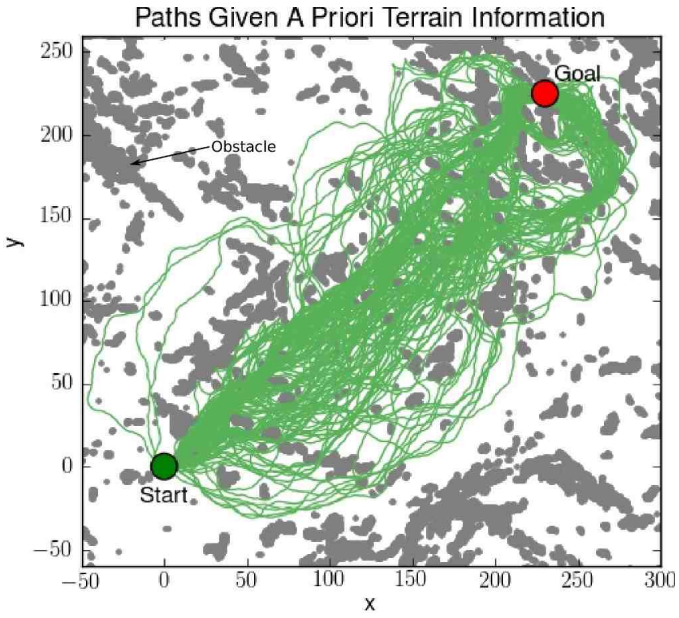


**Figure 4.12:** *The a priori path plans generated when the planner had access to all the terrain data. The plans are not all the same because an RRT-based planner was used.*

### 4.4.2 Simulation Results

On 25 terrain maps, we carried out a total of 6077 simulations of the benchmark, the network-of-reusable-paths navigation system, and the planner with the a priori terrain data. In an individual test of the benchmark or NRP systems, the robot was permitted to take up to 350 scans before it gave up and the scenario timed out. A summary of whether the robot found the goal, got stuck or exceeded the maximum iterations, is shown in Table 4.1. The first thing that is apparent is that the NRP approach reports that it had found the goal in almost every scenario (only 0.8% of all cases exceed the maximum allowable iterations). The benchmark, by contrast, often gets stuck (in 18.5% of all cases) or exceeds 350 iterations (in 5% of all cases). From the perspective of robustness of finding the goal, the NRP system performed better.

We compare two performance metrics in Table 4.2. The first is the localization error at the end of the traverse. This is measured as the distance between the estimated goal location and the actual goal location. The NRP approach is substantially better in this regard, incurring, on average, only 16% the localization error of the benchmark. This is because it only accumulates error along the best path found to the goal through the network. It is conceivable that the benchmark system could do better with a better path

**Table 4.1:** *Summary of times the robot found the goal, got stuck, or exceeded the maximum number of iterations (350) for the benchmark and network-of-reusable-paths navigation systems. The rate of each event is given in parentheses. A total of 6077 simulations were carried out.*

| | Found Goal | Stuck | Exceeded Max Iterations |
|---|---|---|---|
| Benchmark | 4649 (0.765) | 1123 (0.185) | 305 (0.05) |
| Network of reusable paths | 6029 (0.992) | 0 (0.0) | 49 (0.008) |

**Table 4.2:** *A summary of the performance metrics: localization error and number of terrain scans. A terrain scan is acquired when the robot stops (due to encountering an obstacle on the path) and gathers a scan of the surrounding area. The number of terrain scans is calculated in several ways: (i) completely ignoring the cases where the the benchmark got stuck, (ii) the benchmark statistics are calculated using the number of scans the robot took before getting stuck, and (iii) assuming benchmark cases where the robot got stuck as having the maximum number of scans (350). The number of scans collected for the network-of-reusable-paths system is calculated the same in each case and these values are consistently lower than the benchmark system.*

| | Localization error (pixels) | Number of terrain scans (i) | Number of terrain scans (ii) | Number of terrain scans (iii) |
|---|---|---|---|---|
| Benchmark | $102.42 \pm 102.42$ | $95.16 \pm 61.33$ | $86.75 \pm 50.4$ | $143.81 \pm 69.16$ |
| Network of reusable paths | $16.4 \pm 16.4$ | $59.47 \pm 34.9$ | $59.47 \pm 34.9$ | $59.47 \pm 34.9$ |
| Ratio (NRP/Benchmark) | 0.16 | 0.62 | 0.69 | 0.41 |

planner, localization system or terrain assessment, but the NRP system uses the same basic components, and only adds the ability to repeat paths. A summary of localization errors, for many simulations on a single terrain map, is shown in Figure 4.13. The NRP tests (green) are much closer to the actual goal (red) than those of the benchmark (blue).

Two examples of the comparison between the NRP and the benchmark system are shown in Figure 4.14. The top is an example in simple terrain and it is used purely for illustrative purposes. This test is not included in the results. The bottom comparison is representative of the simulations carried out. It shows the start and goal locations, and the approximate density of obstacles (in gray). In both cases, the two approaches began navigating in the same way; it was only after detecting obstacles that the plans differed. The localization error at the goal (as seen by the difference between the estimated track

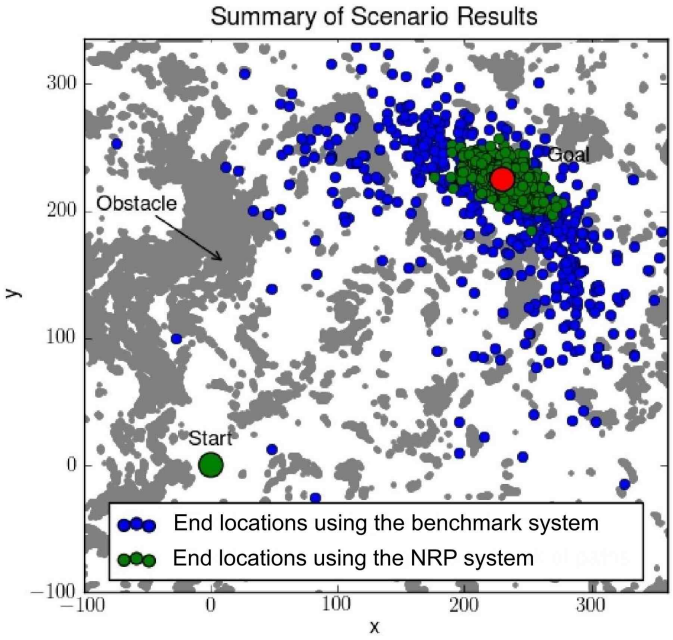


**Figure 4.13:** *Summary of the localization errors for many simulations on a single fractal terrain. These points are the estimated goal locations when the robot believed it was close enough to the goal. The benchmark is in blue and the network-of-reusable-paths approach in green. The goal location is red.*

in blue, and the ground-truth track in gray) was less when using the network-of-reusable-paths approach.

The other metric was the number of terrain scans collected. A terrain scan was acquired when the robot stopped (due to encountering an obstacle on the path) and gathered a scan of the surrounding area. In the benchmark system the resulting assessment data were incrementally added to the global traversability map. In the NRP approach, the resulting local assessment map was attached to the node where the robot collected the scan. The fact that the benchmark system often got stuck, while the NRP approach did not, makes a direct comparison of this metric somewhat less obvious. We have therefore shown it in three different ways where we (i) compare the number of terrain scans when completely ignoring the cases where the benchmark system became stuck, (ii) use the number of scans the benchmark system acquired before becoming stuck, and (iii) assume that when the benchmark system got stuck, it took the maximum number of local assessment maps (350). The number of terrain scans collected by the NRP system was calculated in the same way in all cases. The NRP approach was better in this

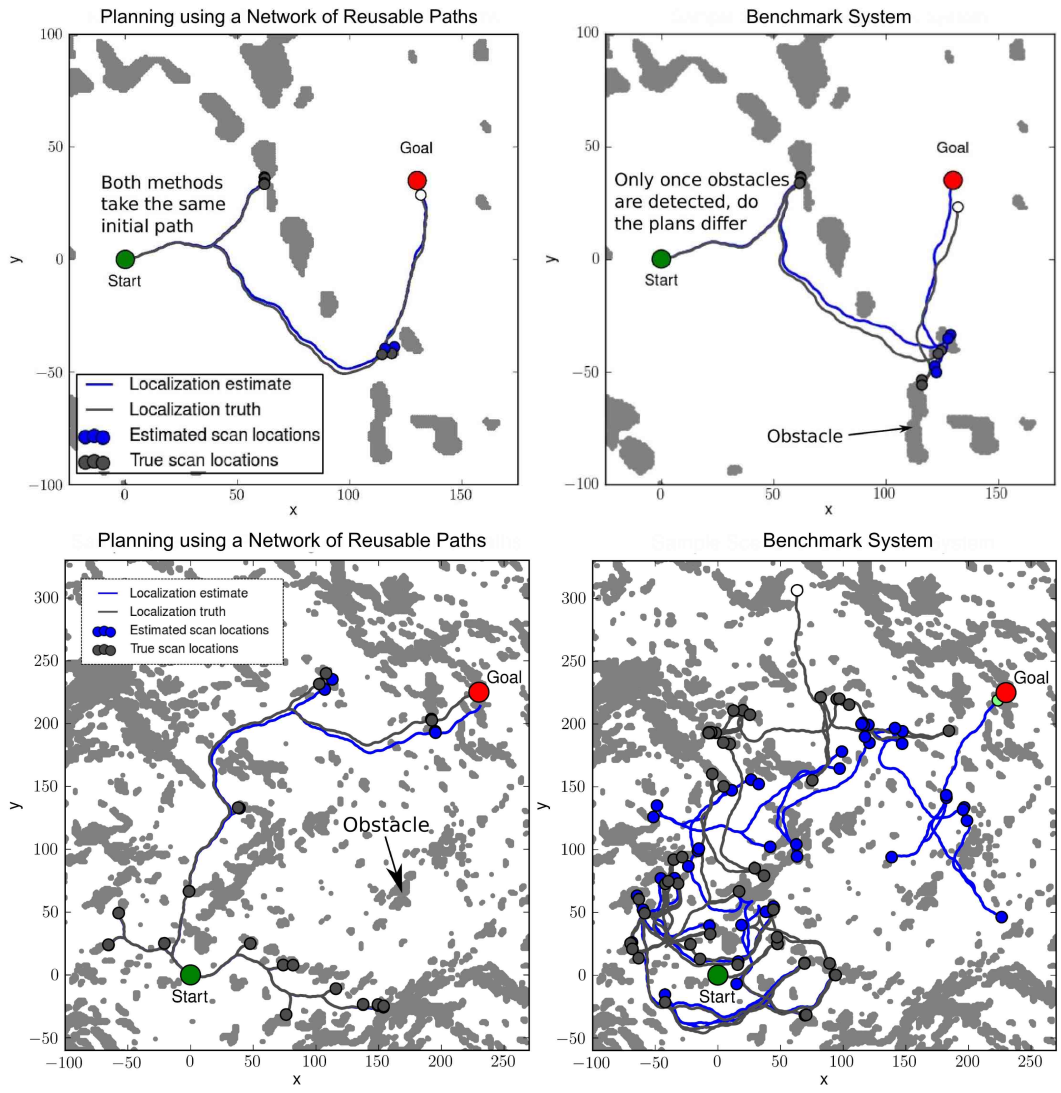


**Figure 4.14:** *A comparison between the benchmark and network-of-reusable-reusable-paths approaches in simple terrain (top) and cluttered terrain (bottom). The NRP approach (left) is an improvement over the benchmark (right) when comparing the number of scans and the localization error at the end of the path.*

measure as well, with between 41% to 69% the number of local assessment maps of the benchmark, depending on the calculation method.

Table 4.3 shows the resulting path lengths for the a priori planner and the two test systems. These are the distances along which the robot accumulated localization error. The a priori paths and the benchmark lengths are simply the distance the robot traveled, while the NRP lengths are the distances along the final start-to-goal paths through the network. The NRP approach results in distances that are nearly the same as that of the

**Table 4.3:** *The distances along which the robot accumulated localization error for the a priori planner as well as the benchmark and NRP systems.*

| | Path length (pixels) |
|---|---|
| Benchmark | $2499.9 \pm 113.2$ |
| Network of reusable paths | $576.5 \pm 45.3$ |
| Using a priori terrain data | $560.8 \pm 140.8$ |

a priori planner (within error bounds). As, expected, it is as if the NRP system takes the a priori path, without having access to the a priori terrain data. We expect that given an optimal planner, we would find that the NRP approach takes the *optimal* path without access to a priori terrain data. This is an area of further research (Section 5.4).

## 4.5 NRP Development Systems

The field systems are presented in the next chapter, but during the development of those systems, we also tested and refined NRP on other robotic platforms. The first used a Pioneer 3-AT robot using a Hokuyo laser rangefinder for a simple bumper. Dead-reckoning was done using wheel odometry and a Vicon motion capture system was used for the teach-and-repeat capability. This system is shown in Figure 4.15, and the experiments done with this system[2] have been presented by Stenning and Barfoot [59].

Figure 4.16 shows the Pioneer 3-AT equipped with a stereo camera[3]. This was the second iteration of the NRP software and it incorporated many lessons from the first test system. This software is very similar to that used in the field trials that have followed.

## 4.6 Summary

This chapter has presented the development of a network of reusable paths, and a path planner that can choose to use reuse previously driven paths. Through theory and simulation we have shown many of the benefits of the NRP approach that is built upon

[2]A video of one of these tests is available at: http://youtu.be/C0J5lWX3xnM
[3]A video of this system is available at: http://youtu.be/ULwo3pX5f4M

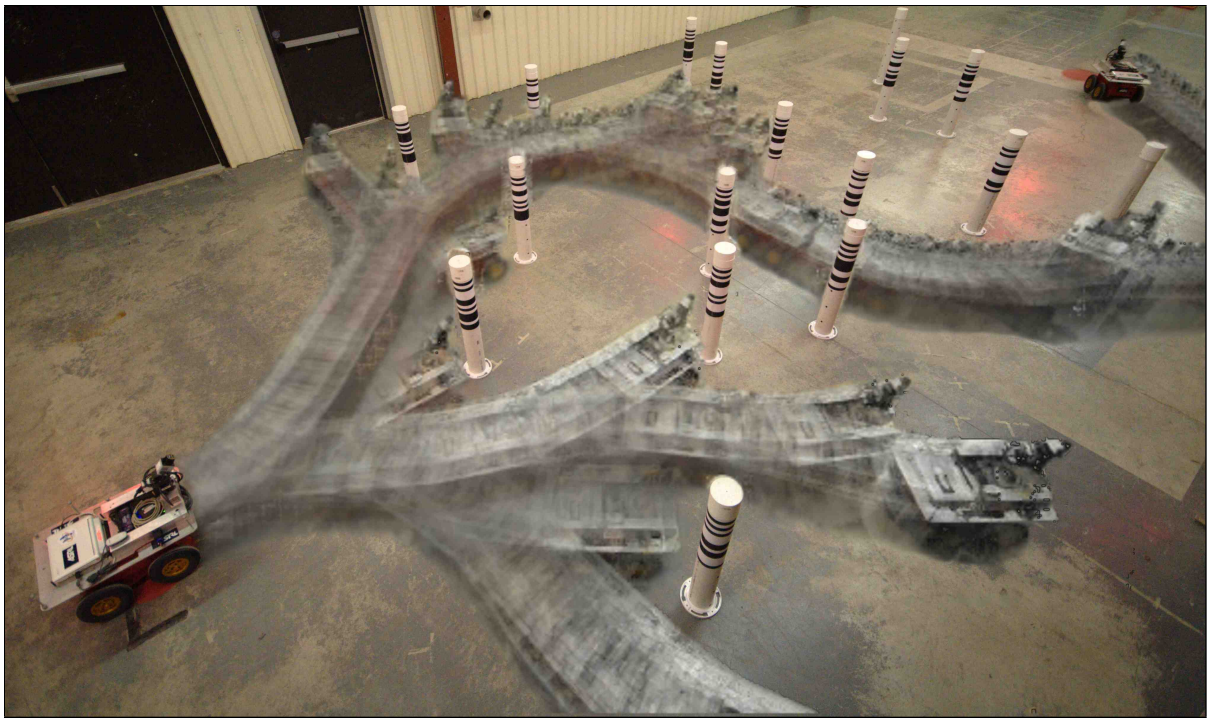

**Figure 4.15:** *A time-lapse view of the first NRP robot seeking a goal (red dot on right) from the start (robot in bottom left). This system, and the associated experiments, are described by Stenning and Barfoot [59].*

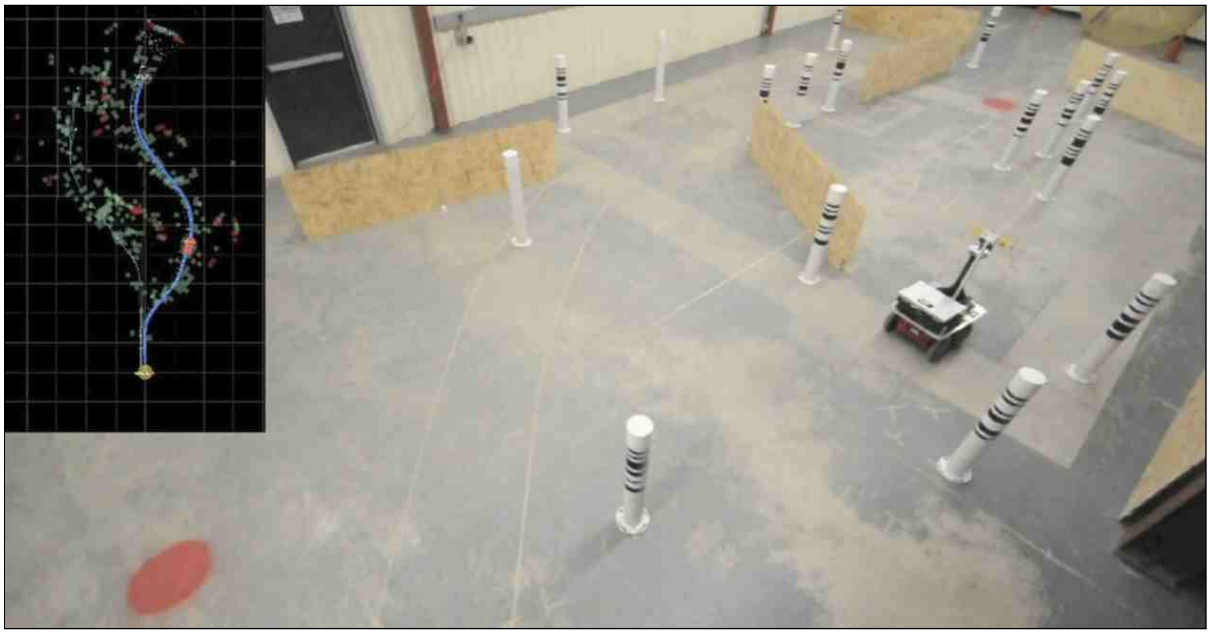

**Figure 4.16:** *The stereo-camera-equipped Pioneer 3-AT robot. This robot was used as a development platform for the field robots that are presented in the next chapter.*

two assumptions (see Section 4.3.1): (i) being able to detect and stop before hazards, and (ii) being able to repeat a previously driven path. The next chapters show how a robot can use a network of reusable paths in outdoor, unstructured terrain.

We believe that growing a network of reusable paths as the map substrate in a navigation framework represents a fundamental paradigm shift within mobile robotics that will allow computationally efficient, accurate, and robust performance in real-world conditions.

# Chapter 5

# A Network of Reusable Paths in the Field

Two sets of field tests were carried out with a robot using a network of reusable paths (the locations are shown on the map in Figure 1.4). The first set used a robot equipped with a stereo camera as the primary sensor, and the second set used a robot equipped with a high-framerate lidar. These tests were carried out in conjunction with mock Lunar exploration mission scenarios conducted in the Sudbury impact crater in Canada [62, 66] and the Mistastin impact structure in Northern Labrador, Canada [64, 65]. Chapter 6 discusses how NRP was used in those missions.

Section 5.1 describes the stereo-camera-equipped robot. The architecture of the GN&C system was common to the two robot configurations, this is discussed in Section 5.1.1. The test setup is presented in Section 5.1.2 before giving the results from the stereo-camera-based tests in Section 5.1.3. The high-framerate lidar system and the corresponding results are presented in Sections 5.2 and 5.2.1, respectively.

## 5.1 The Stereo-Camera-Equipped Robot

The robot, equipped with a stereo camera, is shown in Figure 5.1. The robot base, a robuROC6 made by Robosoft SA, is approximately 1.8 m long and 0.9 m wide. The primary guidance and navigation sensor is a stereo camera that is on the front of the


**Figure 5.1:** *The field robot equipped with a stereo camera. The robot base, a robuROC6 made by Robosoft SA, is approximately* 1.8 m *long,* 0.9 m *wide. The Point Grey Bumblebee XB3 stereo camera is* 1.1 m *above the ground. There are inclinometers in each of passively articulated body segments. Differential GPS was used for the ground-truth localization, the base station was located at the base camp.*

vehicle, about 1.1 m above the ground. The stereo camera was used for the visual-teach-and-repeat system that was the second version of that presented by Furgale and Barfoot [2]. The sparse feature point cloud (i.e., the visual landmarks) was also used by the terrain-assessment system to identify areas that were not safely traversable. There were inclinometers in the three body segments of the robot. These body segments could passively articulate relative to each other to allow the system to conform to the terrain. The differential GPS was used only for the ground-truth localization and was not used for any of the autonomous navigation.

### 5.1.1 The Guidance, Navigation, and Control System

The high-level guidance, navigation, and control (GN&C) architecture is shown in Figure 5.2. In this architecture, we can see all the elements present in the generic GN&C diagram shown in Figure 1.3. The Graph Navigation element is the visual-teach-and-

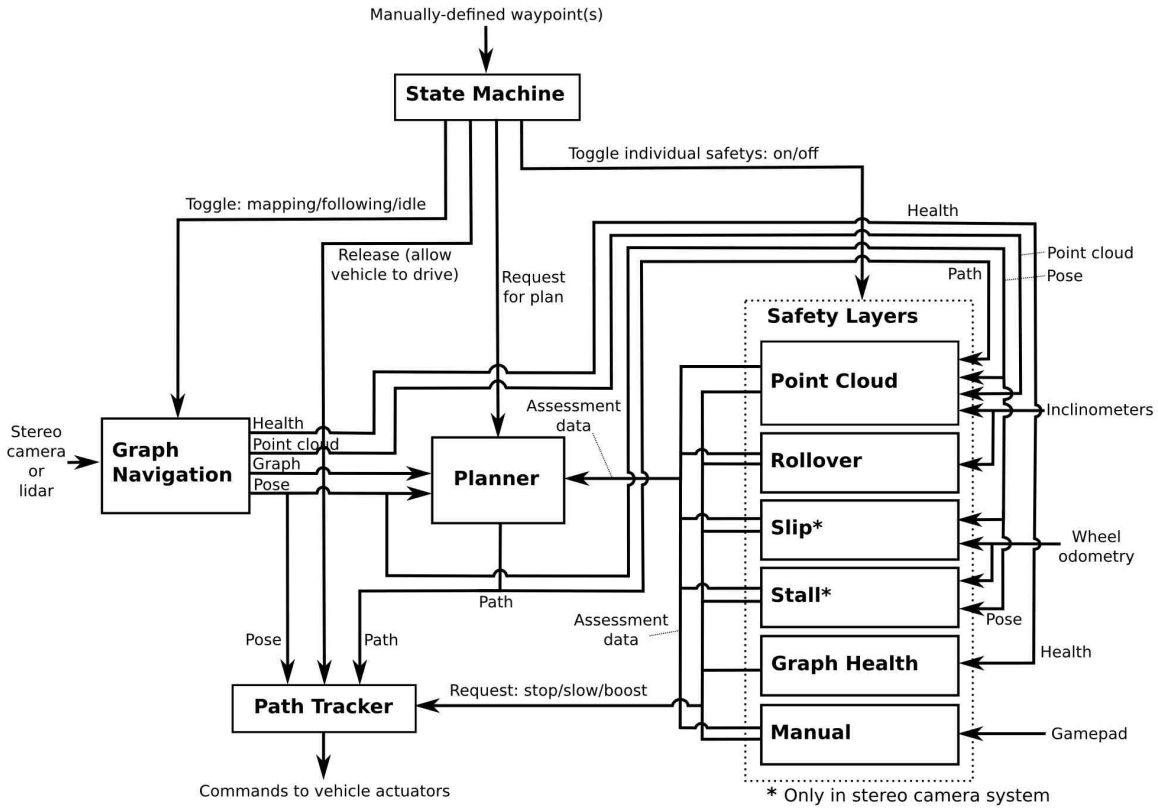


**Figure 5.2:** *GN&C architecture of the field robots that makes use of a network of reusable paths. The safety layers were turned on by the state machine, as appropriate. These safety layers provided local terrain assessment data to the planner, and could ask the vehicle to stop, slow down, or even try to boost the speed (in the case of a stall). The Graph Navigation system maintained the graph structure of the network of reusable paths. These data were shared with the planner. The planner created a plan based on the current network and the assessment data stored at the nodes of the network.*

repeat system for use on a network of reusable paths (rather than a single path). It carried out localization and mapping by adding new local maps to the graph as the robot drove into new terrain, and it reported the pose of the vehicle relative to a node in the graph (the frame in which localization is reported often changed to best suit the task), at approximately 8-10 Hz in the stereo-camera-based system. The path tracker used the current pose to track the plan. The speed of the robot varied from 0.25 m/s to 1.0 m/s, depending on the roughness terrain and the curvature of the path.

The safety layers had two main roles: (i) provide terrain assessment data to the planner and (ii) ensure that the vehicle was safe as it drove. Not all safety layers were active at all times, the Point Cloud, Rollover, Slip, Stall and Manual safety layers were

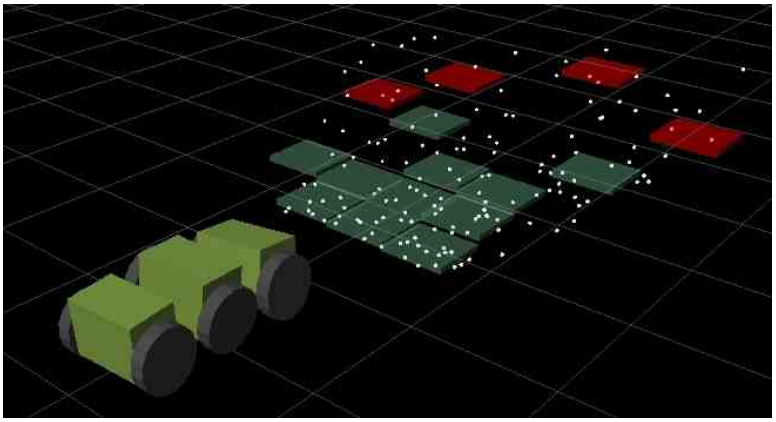

**Figure 5.3:** *An Rviz visualization of an example sparse-stereo point cloud and the resulting first-phase terrain assessment. The point cloud was from the point features used for stereo-based visual teach and repeat. Cells identified in red have been flagged as containing strongly vertical terrain. The green cells show areas that appeared to be locally flat, and their mean height was recorded. These data were used in the second phase of the assessment.*

active only when the robot was driving into new terrain (adding to the network, teaching). The Graph Health layer was active only when the robot was repeating a previous part of the network.

The Point Cloud safety layer looked ahead on the path to detect potential geometric hazards. This terrain assessment was done in two phases. The first phase took place immediately after the point cloud was captured. Rviz, a visualization environment provided by Robot Operating System (ROS) [124], was used to visualize a point cloud and the resulting first-phase assessment from the stereo camera in Figure 5.3. ROS was used extensively in these systems. The point cloud was transformed into a gravity-aligned frame (i.e., the vector aligned with the acceleration due to gravity was in the negative $z$-direction) with the $x$-direction straight in front of the vehicle. The cloud was then divided into square cells that were significantly smaller than the vehicle footprint (in this case, 0.5 m by 0.5 m). Cells that contained a large change in elevation were robustly identified and flagged (they are marked red in the Rviz visualization). The process was quite similar to that used by Thrun et al. [23]. Cells that were not flagged appeared to be locally flat and the mean height of the points was recorded. If there were not sufficient data in a cell (this system required a minimum of five points), then no check was made (note how some areas of the cloud do not have a corresponding green or red cell). These

assessment data were forwarded to the path planner where they were available to be used in the second phase of assessment when planning later paths.

The second phase of the assessment was done on demand, meaning, only for certain potential movements. The safety layer used the second phase assessment to look forward along the current path, and it would stop the vehicle if a hazard was detected. The planner used the second phase terrain assessment on candidate path segments. The second phase fitted a plane to the cells in the robot's footprint and then slid that footprint along a candidate path segment. The segment was considered to be not viable if at any point the footprint contained a flagged cell. If there were no flagged cells in a footprint, a plane was then fit to the mean heights and the $x$- and $y$- positions of the cells. The magnitudes of the roll, pitch, step, and roughness were then checked against a set of thresholds. If any measure exceeded the threshold, the segment was not viable (this is similar to the Morphin algorithm [67]). If the Point Cloud safety layer detected a segment that was not viable, it stopped the vehicle, and if the robot had not reached the goal, a new plan was created. The Point Cloud safety layer would also slow the vehicle in challenging areas that did not quite meet the obstacle thresholds.

The other safety layers (with the exception of the Graph Health) can be conceptualized as bumpers. Triggering a bumper would send an obstacle assessment to the planner and stop the path tracker. The Rollover layer monitored the inclinometers in each pod to ensure the vehicle did not tip. It also protected against self-collision (so that payloads did not hit each other). The Rollover layer would slow the vehicle in challenging areas where obstacle thresholds were not met. The Slip layer would trigger when the relative motion estimates from visual odometry and wheel odometry differed significantly. The Stall layer would trigger when neither visual odometry nor wheel odometry believed there was sufficient motion given the control inputs. The stall condition first triggered an attempt to boost the speed of the path tracker (provide a bit more power to the wheels). It added an obstacle to the map if the boost was unsuccessful. The Manual layer allowed a human observer to insert an obstacle into the network at the current position. This was necessary for certain classes of obstacle that could not reliably detected (e.g., overhangs that were above the camera field of view, but that could still hit payloads on the vehicle).

It can be thought of as a placeholder for more advanced hazard detection systems.

The Graph Health layer controlled only the speed of the robot, it would slow the vehicle if Graph Navigation was having difficulty localizing to the map. Reducing the speed would reduce motion blur in the images. If the vehicle drove along the previous path for too far without localizing against the map, the Graph Health layer would stop the robot and continue to attempt to localize without running the risk of driving into unsafe terrain. The vehicle was allowed to continue once the localization was reacquired. In practice, this also allowed human operators the opportunity to return the robot to the path manually.

### 5.1.2 Test Description

Individual tests were started from various initial poses in order to have the robot encounter a variety of terrain. Waypoints (goals) were defined as points relative to the initial position (the first node in the network). Two variations were used: (i) the robot had a single distant waypoint and a second waypoint at the initial position (to cause the robot to return after finding the goal), and for the stereo-camera-equipped robot, (ii) five distant waypoints were given, plus a sixth waypoint at the initial pose (again, to return to the start after visiting the goals).

### 5.1.3 Results from Stereo-Camera-Based NRP

The results of using the stereo-camera-equipped robot are given in Tables 5.1, 5.2 and 5.3. Table 5.3 gives a summary of all the tests. The robot drove a total of 11.10 km, while creating 4.79 km of networks (43.2 %). This means that 6.31 km (56.8 %) of driving was repeating previous paths.

Figures 5.4, 5.5 and 5.6 take a closer look at test 21. Videos of test 9[1] and test 21[2] are available online. In test 21, the goal was set 62 m straight ahead of the start position and the robot had no prior knowledge of the terrain. The robot autonomously found the distant waypoint and then returned to the start position. Figure 5.4 gives an overview

[1] A video of test 9 is available at: http://youtu.be/
[2] A video of test 21 is available at: http://youtu.be/0NmpSBA1XQM

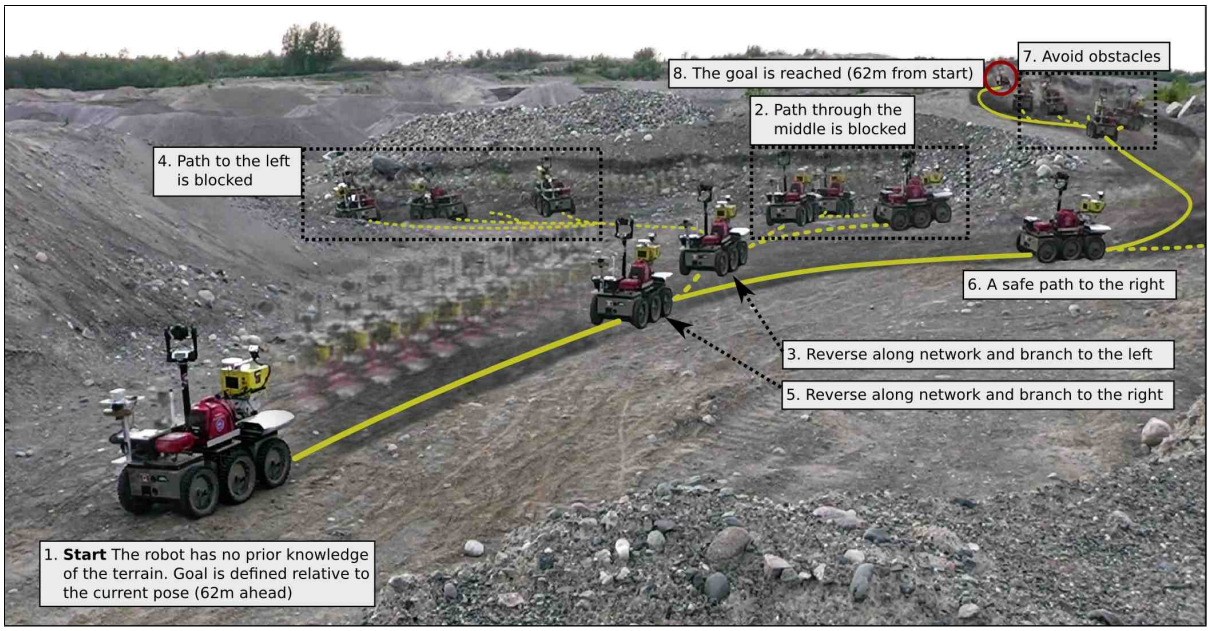


**Figure 5.4:** *The sequence of events in test 21 for the stereo-camera-equipped robot. The distant waypoint was set 62 m straight ahead of the initial starting position. The robot had no prior knowledge of the terrain. Initially it tried to go straight to the goal, it then planned several paths that went to the left before it found that way, too, was blocked. The robot then found the path to the right and after avoiding a few more obstacles it reached the waypoint and then returned to the start position, all completely autonomously.*

of the test, showing the terrain and the network of reusable paths.

Figure 5.5 gives a snapshot, as an Rviz visualization, into the internal state of the system as it was seeking the distant waypoint. We can see the path that the robot was following when it detected an obstacle ahead (the thick blue line is the path that reused the existing network, the thick red line is the path into new terrain). Also shown is the state of the RRT tree (gray) with the best path found so far (yellow). The network, in the stereo camera frame, is shown in light blue along with a selection of node identification numbers.

A top-down view of test 21 is shown in Figure 5.6. This figure emphasizes just how sparse the terrain data were, yet the robot still reached the distant waypoint. The vehicle legend in the bottom right corner shows the field of view of the stereo-camera-based Point Cloud safety layer, along with the minimum turning radius of the vehicle. The remarkable thing is that the system worked reliably, even though the obstacle detection range was very short in relation to the vehicle maneuverability, and the assessment data were very

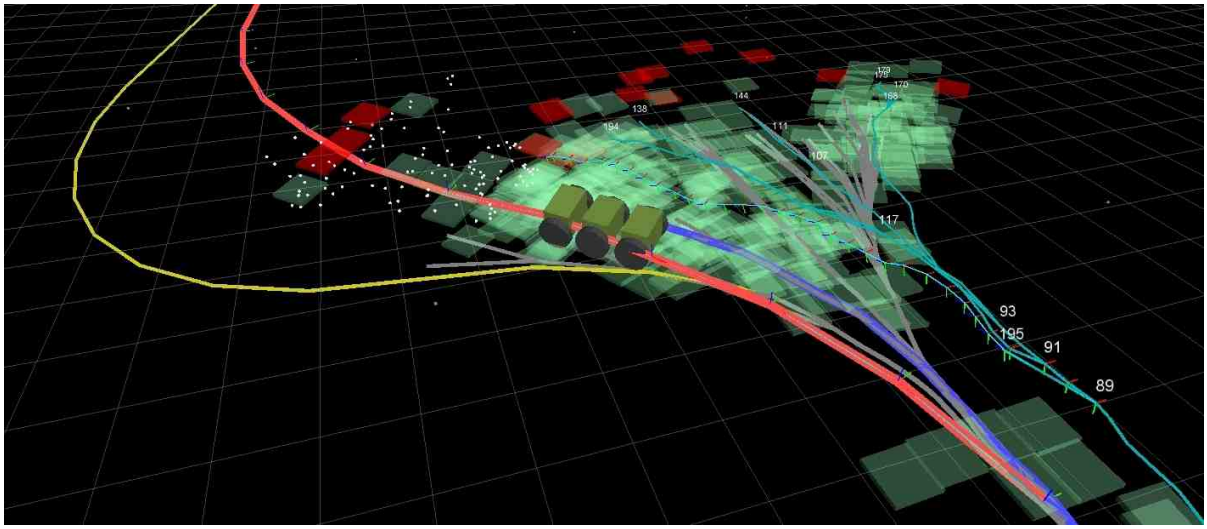


**Figure 5.5:** *An Rviz visualization of the internal state of the rover localization and guidance systems during test 21. The thick blue and red lines are the previous plan (i.e., repeat existing network, the blue section, and drive forward into new terrain, along the red section). However, an obstacle was detected (see the assessment overlaying the most recent point cloud) along the red path so the vehicle stopped and the robot is now re-planning. The yellow path is the best path found so far. The gray tree is the current state of the RRT. The network is drawn in light blue with node identification numbers drawn above branch points and at the ends of branches.*

sparse. In this test, the robot encountered 28 obstacles (17 ranged and 11 rollover). For all the tests that were carried out with this system, the average range of a Point Cloud safety layer obstacle was 3.4 m, and the max range was 5 m. This figure also shows how we measured the values in Tables 5.1-5.3.

Table 5.1 provides the results from attempting to reach a single distant waypoint in unknown terrain and then returning to the start pose (the initial node in the network). The *range-to-waypoint* is the absolute distance to the distant waypoint from the start point. The *estimated-final-range-to-waypoint* and the *actual-final-range-to-waypoint* are respectively the robot's estimated and actual range to the distant waypoint when either the robot thought it was within the capture threshold (25 m for tests 1-7, 10 m for tests 8-13, and 7 m for tests 14-23) or when the waypoint was aborted. The *distance-to-goal-from-start-on-network* is the distance along the network from the waypoint capture or abort point, plus the final distance to the goal. The *total-network-length* is the cumulative distance of all the paths in the network, and the *total-distance-traveled* is the total distance the robot drove, including the return to the start. The last two columns

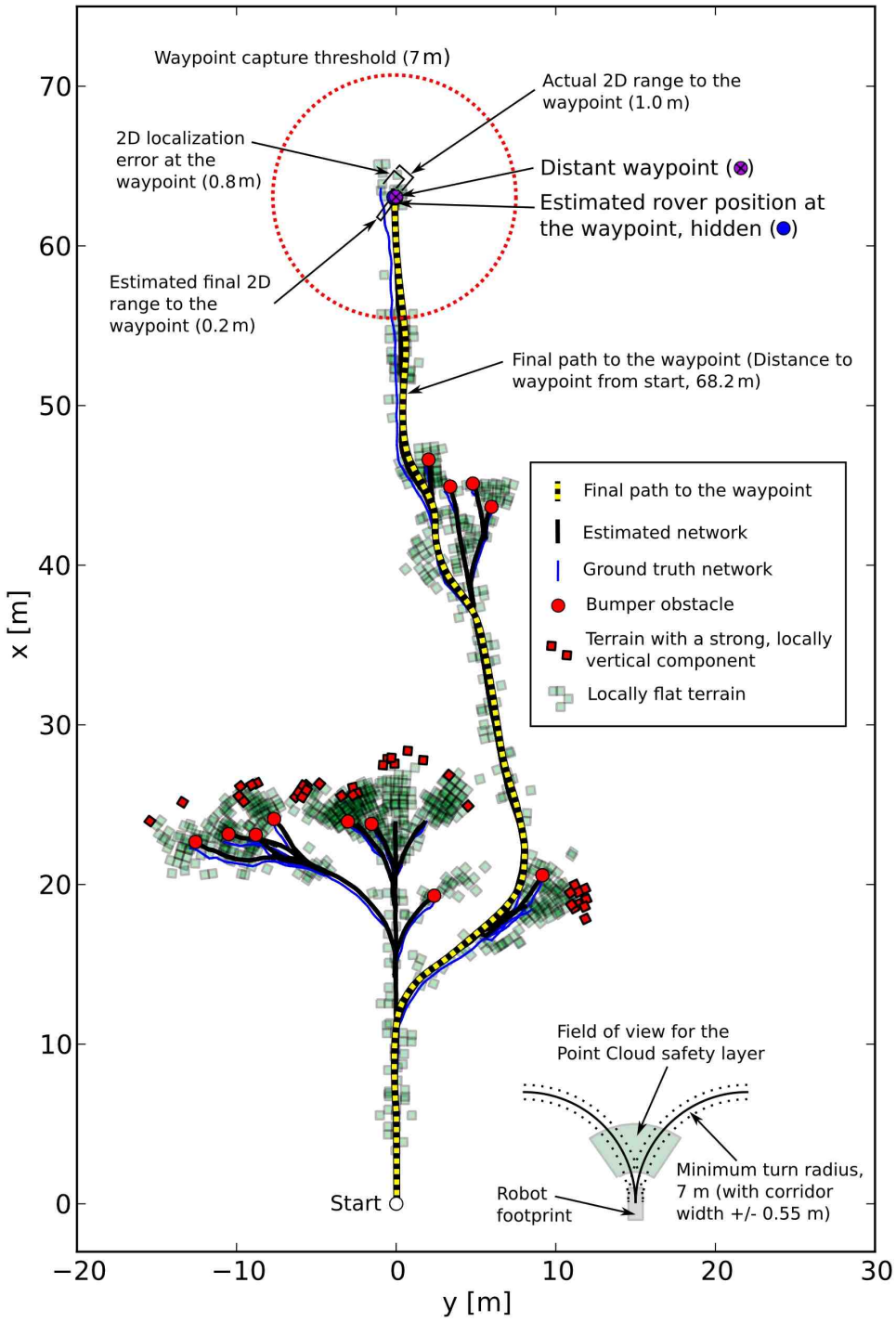


**Figure 5.6:** *An overhead view of test 21 for the stereo-camera-equipped robot. Along the way the robot encountered 28 obstacles (17 ranged, 11 rollover). The average ranged-obstacle detection range was 3.4 m, the maximum range was 5 m. These ranges are significantly less than the minimum turning radius of the vehicle. There are not many terrain assessment data that are collected by the robot, yet the robot autonomously reached the goal and then returned to the start. See the supplementary media for a movie of this test.*

**Table 5.1:** *Results of tests 1-21 where the stereo-camera-equipped mobile robot used NRP to seek a single waypoint and return to the start. The* range-to-waypoint *is the absolute distance to the distant waypoint from the start point. The* estimated-final-range-to-waypoint *and the* actual-final-range-to-waypoint *are respectively the robot's estimated and actual range to the distant waypoint when either the robot thought it was within the capture threshold or when the waypoint was aborted. The* distance-to-waypoint-from-start-on-network *is the distance along the network from the waypoint capture or abort point, plus the final distance to the goal. The* total-network-length *is the cumulative distance of all the paths in the network, and the* total-distance-traveled *is the total distance the robot drove, including the return to the start. The last two columns indicate if the robot believed it had reached the distant waypoint and if the robot successfully returned to the start. Two sets of statistics are given. The first is for all 21 tests, the second is for only the 18 tests where the robot believed it had reached the distant waypoint. Tests 9 and 21 (bold) are shown in videos in the supplementary media.*

| Test | Range to waypoint (m) | Estimated final range to waypoint (m) | Actual final range to waypoint (m) | Final localization error (m) | Distance to waypoint from start on network (m) | Total network length (m) | Total distance traveled (m) | Reached waypoint? | Returned to start? |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 90.1 | 10.9 | 5.6 | 7.4 | 92.5 | 235.2 | 496.9 | Yes | Yes |
| 2 | 133.1 | 22.9 | 20.7 | 2.9 | 118.4 | 180.4 | 259.9 | Yes | Yes |
| 3 | 73.8 | 10.3 | 3.8 | 8.5 | 83.6 | 102.9 | 220.1 | Yes | Yes |
| 4 | 129.4 | 64.4 | 55.1 | 11.9 | 143.6 | 200.3 | 557.9 | **No** | Yes |
| 5 | 200.0 | 24.9 | 12.1 | 19.7 | 235.5 | 568.2 | 1,293.0 | Yes | Yes |
| 6 | 206.9 | 111.7 | 99.1 | 12.8 | 103.1 | 240.7 | 542.4 | **No** | Yes |
| 7 | 230.5 | 149.8 | 142.8 | 8.3 | 303.9 | 543.6 | 1,364.7 | **No** | Yes |
| 8 | 73.5 | 9.0 | 5.3 | 7.7 | 75.6 | 129.2 | 270.0 | Yes | Yes |
| **9** | **80.0** | **9.1** | **6.6** | **2.7** | **94.4** | **162.8** | **338.4** | **Yes** | **Yes** |
| 10 | 73.8 | 7.0 | 3.6 | 5.6 | 84.4 | 118.6 | 246.3 | Yes | Yes |
| 11 | 82.0 | 7.7 | 5.1 | 2.4 | 86.5 | 109.2 | 222.0 | Yes | Yes |
| 12 | 72.2 | 4.2 | 2.9 | 1.4 | 85.4 | 81.4 | 164.8 | Yes | Yes |
| 13 | 54.9 | 9.8 | 9.0 | 1.5 | 50.3 | 50.3 | 101.1 | Yes | Yes |
| 14 | 103.8 | 1.3 | 4.5 | 4.6 | 111.6 | 149.1 | 319.7 | Yes | Yes |
| 15 | 114.9 | 0.9 | 4.2 | 5.0 | 147.2 | 248.9 | 523.5 | Yes | Yes |
| 16 | 65.8 | 1.4 | 0.9 | 2.3 | 70.8 | 131.6 | 273.8 | Yes | Yes |
| 17 | 44.0 | 6.2 | 5.7 | 0.5 | 48.8 | 109.3 | 228.7 | Yes | Yes |
| 18 | 50.0 | 2.6 | 3.2 | 1.0 | 65.4 | 139.4 | 390.4 | Yes | Yes |
| 19 | 67.0 | 6.5 | 6.5 | 0.5 | 70.4 | 144.1 | 298.5 | Yes | Yes |
| 20 | 67.0 | 6.8 | 6.5 | 0.2 | 71.2 | 143.0 | 298.5 | Yes | Yes |
| **21** | **62.0** | **0.2** | **1.0** | **0.8** | **68.2** | **187.1** | **408.1** | **Yes** | **Yes** |
| Total | 2,075.7 | 467.5 | 404.4 | 107.8 | 2,210.7 | 3,975.5 | 8,818.8 | 18/21 | 21/21 |
| Mean | 98.8 | 22.3 | 19.3 | 5.1 | 105.3 | 189.3 | 419.9 | 0.857 | 1.000 |
| Total when reached goal | 1,508.9 | 141.6 | 107.4 | 74.7 | 1,660.1 | 2,990.8 | 6,353.7 | 18/18 | 18/18 |
| Mean when reached goal | 83.8 | 7.9 | 6.0 | 4.1 | 92.2 | 166.2 | 353.0 | 1.000 | 1.000 |

**Table 5.2:** *Results of a stereo-camera-equipped mobile robot on a network of reusable paths attempting to seek five waypoints and return to the start. Tests 22 and 23 had the robot navigate to five distant waypoints in unknown terrain, and then return to the start position. All waypoints were reached and the robot returned to the start in both tests. The columns are as in Table 5.1.*

| Test | Range to waypoint (m) | Estimated final range to waypoint (m) | Actual final range to waypoint (m) | Final localization error (m) | Distance to waypoint from start on network (m) | Total network length (m) | Total distance traveled (m) | Reached waypoint? | Returned to start? |
|---|---|---|---|---|---|---|---|---|---|
| 22 | | | | | | 345.1 | 701.5 | | |
| a | 72.2 | 2.2 | 2.1 | 0.1 | 85.0 | | | Yes | Yes |
| b | 54.9 | 0.3 | 1.6 | 1.3 | 84.3 | | | Yes | Yes |
| c | 103.8 | 1.2 | 4.6 | 5.8 | 148.4 | | | Yes | Yes |
| d | 114.9 | 0.9 | 7.5 | 7.9 | 174.7 | | | Yes | Yes |
| e | 65.8 | 3.6 | 1.3 | 2.3 | 72.8 | | | Yes | Yes |
| 23 | | | | | | 472.5 | 1,584.1 | | |
| a | 54.9 | 0.7 | 1.6 | 1.1 | 62.2 | | | Yes | Yes |
| b | 65.8 | 1.5 | 1.8 | 2.1 | 76.1 | | | Yes | Yes |
| c | 114.9 | 1.1 | 4.4 | 5.5 | 170.5 | | | Yes | Yes |
| d | 72.2 | 2.6 | 3.2 | 1.0 | 92.7 | | | Yes | Yes |
| e | 103.8 | 4.4 | 4.8 | 2.9 | 148.3 | | | Yes | Yes |
| Total | 823.1 | 18.5 | 32.9 | 29.9 | 1,115.0 | 817.6 | 2,285.6 | 10 of 10 | 10 of 10 |
| Mean | 26.6 | 0.6 | 1.1 | 1.0 | 36.0 | 408.8 | 1,142.8 | 1.000 | 1.000 |

**Table 5.3:** *Summary of all results of a stereo-camera-equipped mobile robot on a network of reusable paths attempting to autonomously seek waypoints and return to the start. Two sets of means and totals are given. The first is for all 31 distant waypoints, the second is for only the 28 waypoints the robot believed it had reached. The columns are as described in Table 5.1.*

| | Range to waypoint (m) | Estimated final range to waypoint (m) | Actual final range to waypoint (m) | Final localization error (m) | Distance to waypoint from start on network (m) | Total network length (m) | Total distance traveled (m) | Reached waypoint? | Returned to start? |
|---|---|---|---|---|---|---|---|---|---|
| Total | 2,898.8 | 486.0 | 437.3 | 137.6 | 3,325.7 | 4,793.1 | 11,104.4 | 28/31 | 31/31 |
| Mean | 93.5 | 15.7 | 14.1 | 4.4 | 107.3 | 208.4 | 482.8 | 0.903 | 1.000 |
| Total when reached goal | 2,332.0 | 160.1 | 140.3 | 104.5 | 2,775.1 | 3,808.4 | 8,639.3 | 28/28 | 28/28 |
| Mean when reached goal | 83.3 | 5.7 | 5.0 | 3.7 | 99.1 | 190.4 | 432.0 | 1.000 | 1.000 |

indicate if the robot believed it had reached the distant waypoint, and if the robot successfully returned to the start. Of the 21 single waypoints (not including instructions to return to the start), the robot reached 18 (0.857 success rate). In all 21 tests the robot successfully returned to the start. In the cases where the robot failed to reach the waypoint, a human operator made a judgment to return home after letting the attempt to reach the goal continue for an extended period of time. This can be thought of as a timeout on a waypoint. The incomplete waypoints were all physically reachable, but they were distant, and through areas that were difficult for accurate visual odometry (fine, repetitive texture), which led to faster growth of the localization error, and in some cases made the estimated goal position drift into an unreachable place. Two sets of cumulative and mean statistics for the tests are at the end of the table. The first are for all 21 tests, the second are for only those tests where the robot believed it had reached the goal.

Table 5.2 gives the results for the scenario where the robot was given five waypoints (the same points were used in each test, but the sequence was in a different order, see Figure 5.8) and then directed to return to the start. The columns are the same as in Table 5.1. All the waypoints were reached and the robot returned to the start in both tests.

Figure 5.7 shows an overview of test 23. The robot was told to seek five waypoints, in unknown terrain, and then return to the start. The estimated position of the network is the black line. As the network extends farther from the start, the localization drift becomes apparent. The true network position is shown by the blue line. The localization error was rolled back by reversing along a previous path and did not necessarily continue to accumulate as the robot moved to seek new waypoints. The shapes of the networks from tests 22 and 23 are compared in Figure 5.8. In these tests, the same waypoints were visited, but in a different order. We expect that the final path to any individual waypoint would have converged to be the same in both tests, if the NRP planner were optimal. Of course, a practical system may attempt to trade off time for accuracy by penalizing the time taken to repeat a previous path. This is a consideration for later works.

Occasionally, it was necessary to intervene with an experiment. The interventions are listed in Table 5.4. As mentioned in Section 5.1.1, the *obstacle* interventions are a

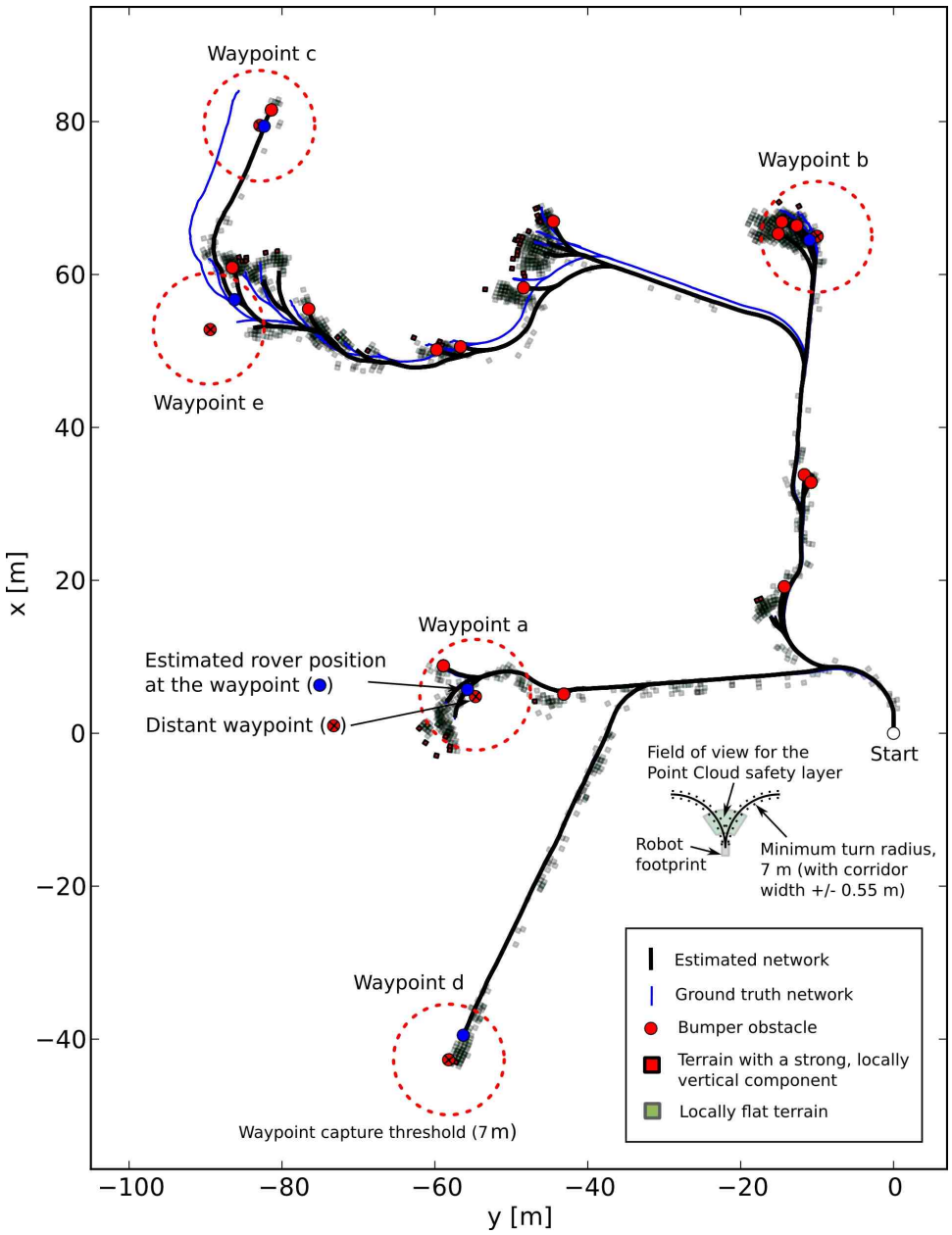


**Figure 5.7:** *An overhead view of test 23, where the robot was told to seek five distant waypoints, in the order a through e, and then return to the start.*

placeholder for a missing method of terrain assessment. Interventions of this type might have been avoided by using more sophisticated terrain assessment algorithms, or sensors with a wider field of view. The other interventions were necessary because the robot was unable to localize against the map while repeating a path (usually because the appearance of the scene had changed from when the path was added to the network). However, in each of these localization failures the robot successfully completed the path on the second

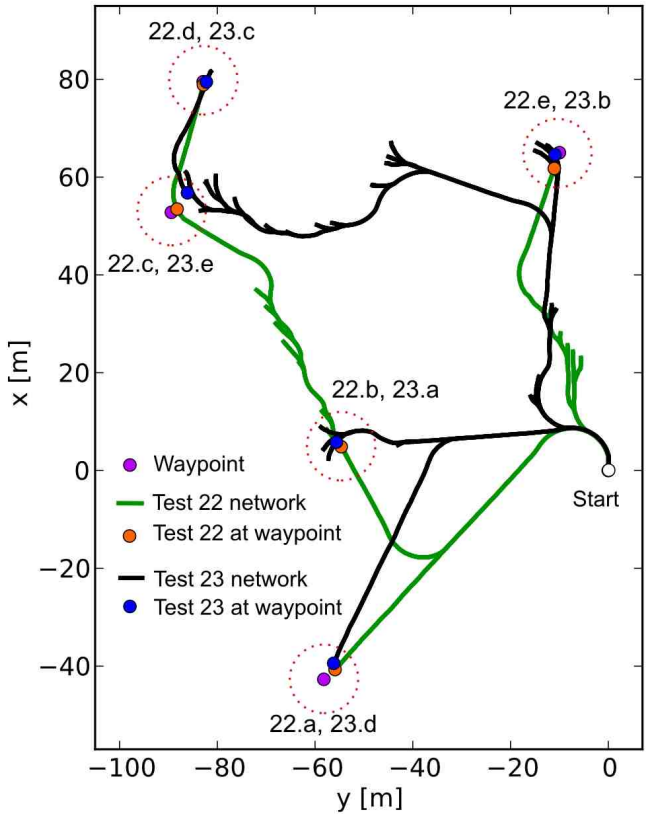


**Figure 5.8:** *The shape of the network in test 22 (green) compared to that of test 23 (black). The two tests had the same set of waypoints, but in different orders. If the planner were optimal we expect that the final path to each waypoint would be the same, regardless of the order in which the robot was told to seek that waypoint.*

attempt (after being manually driven backward along the path). This suggests that if the robot had been able to automatically reverse and re-attempt to drive the path in question, no intervention would have been necessary. We can look to some recent work by Churchill and Newman [125] for one way of providing this capability. They present an approach to lifelong navigation that would aid in building paths that are more robust to appearance changes in the scene. Essentially, one could consider this system to always be automatically teaching new visual landmarks and/or localizing using existing landmarks, as necessary. This capability is a consideration for later works.

Figure 5.9 shows how localization error grows with the distance traveled from the frame in which the localization is reported. The error is measured in the frame at the start of the network (the goal definition frame). The error tends to grow as the robot travels farther from the start. The error is measured in three ways: at the top is the

**Table 5.4:** *Manual interventions in the tests using the stereo-camera-equipped robot. In the* reposition *interventions an operator sent commands to the robot for the purpose stated in the description of the intervention. In the* obstacle *interventions an operator manually added an obstacle at the robot's position.*

| Test number | Intervention type | Description of intervention |
|---|---|---|
| 4 | Reposition | Map localization was lost for just over 3 m (the distance threshold to stop) while repeating a previous path on uniformly textured gravel that had dried out in the sun over the course of the experiment (i.e., the appearance had changed from dark to light and there were not many distinct visual landmarks). The robot stopped but could not match to the map. The operator took control of the robot and returned it to a section where it could localize against the map. The robot then re-attempted, and successfully completed, the difficult section without further intervention. |
| 18 | Reposition | A section of the network had a significant lateral slope. During the course of the experiment the robot repeated this section multiple times (forward and reverse along a 5 m section) and the vehicle accumulated a lateral error of about 50 cm (the slope was likely beyond what the path tracker could handle reliably). The localization failed to match to the visual-landmark map so the robot stopped. The operator manually returned the robot to the path, the localization match was re-acquired, and the experiment continued. |
| 22 | Obstacle | Four manual obstacles were added when the robot came near some large, overhanging heavy equipment (a gravel conveyor that was not in use). |
| 23 | Reposition | The test was carried out in the early evening a few hours before sunset, so the lighting was starting to change. The robot was repeating a previous path but it was having difficulty because the appearance had changed due to the changing lighting conditions. At that point, several people on all-terrain vehicles and dirt bikes approached the robot and entered the frame of the camera and the map localization was lost. The robot stopped and the people moved but the robot could not re-localize. The operator moved the robot to a previous point and let it try the section again, at which point the robot successfully drove the path and no further interventions were necessary. |

absolute range-error in $xyz$, the middle is the absolute range-error in just the $x$- and $y$-dimensions (what we care about most), and the bottom shows the $z$-component of the error. Test 23 is drawn in red to better show the network structure of the test; the network is just mapped into the distance-error space. This emphasizes that the error rolls back as the robot reverses along the path toward the goal definition node. One can speculate how the error would grow if dead reckoning were used throughout; however, this information is not available. With NRP, the error contribution is only due to the dead-reckoning error on the final path to the goal, whereas other (non-SLAM) systems

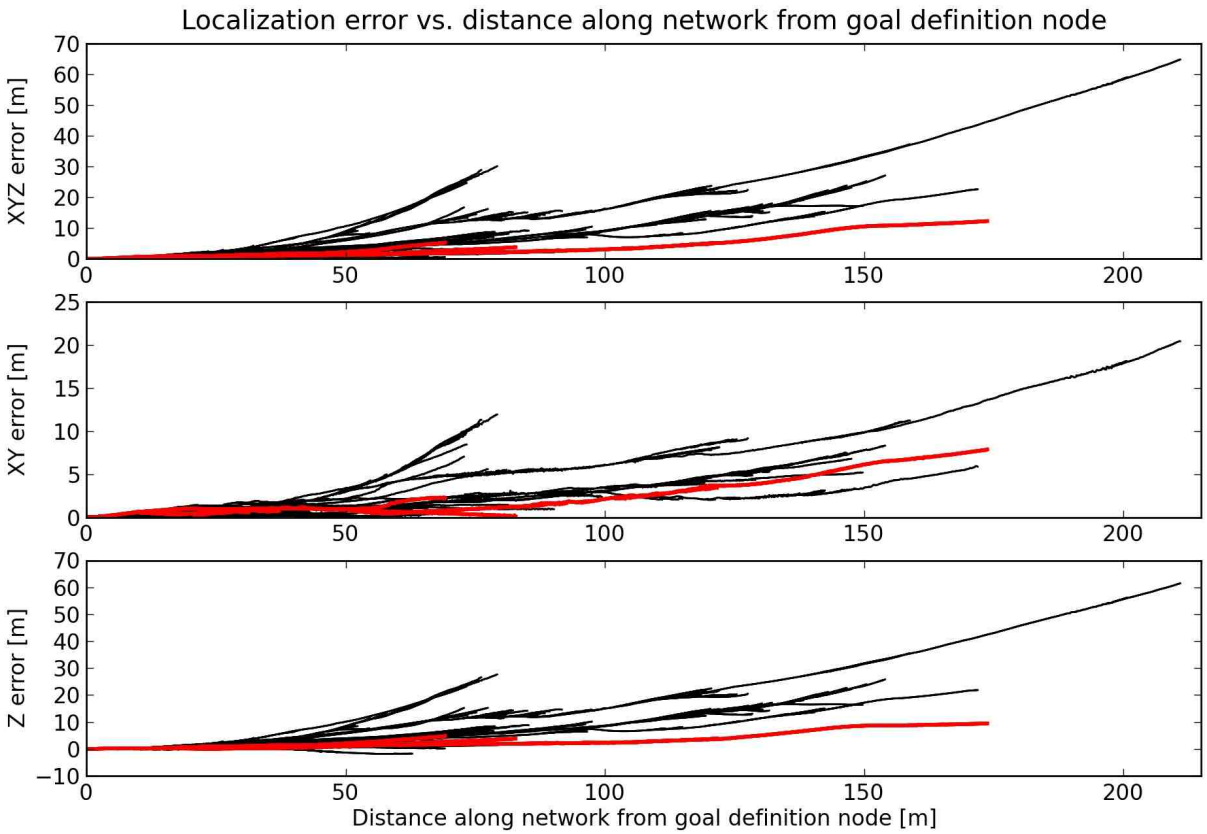


**Figure 5.9:** *Localization error versus distance along the network for tests 1-23 using the stereo-camera-equipped robot. Note the network structure of the of the tests. Test 23 is highlighted in red to better show the detail. The ground-truth localization was measured using differential GPS.*

will accumulate errors along the entire traverse path.

The $z$-component has a significant contribution to the overall localization error. The estimated pitch of the vehicle had a negative bias (e.g., the vehicle tended to estimate that it was pitching upward) whose magnitude seemed strongly correlated to the visual appearance of the terrain that the vehicle was in, making calibration before the field trials unsuitable. Lambert [126] does a deeper analysis of this visual-odometry bias.

## 5.2 The Lidar-Equipped Robot

The stereo-camera-equipped robot was limited to operating in similar lighting conditions throughout a test, because a change in the lighting conditions changed the appearance of the scene and made localizing against the map difficult or impossible. To solve this


**Figure 5.10:** *The Autonosys-equipped robot in the Mistastin impact structure in Northern Labrador, Canada.*

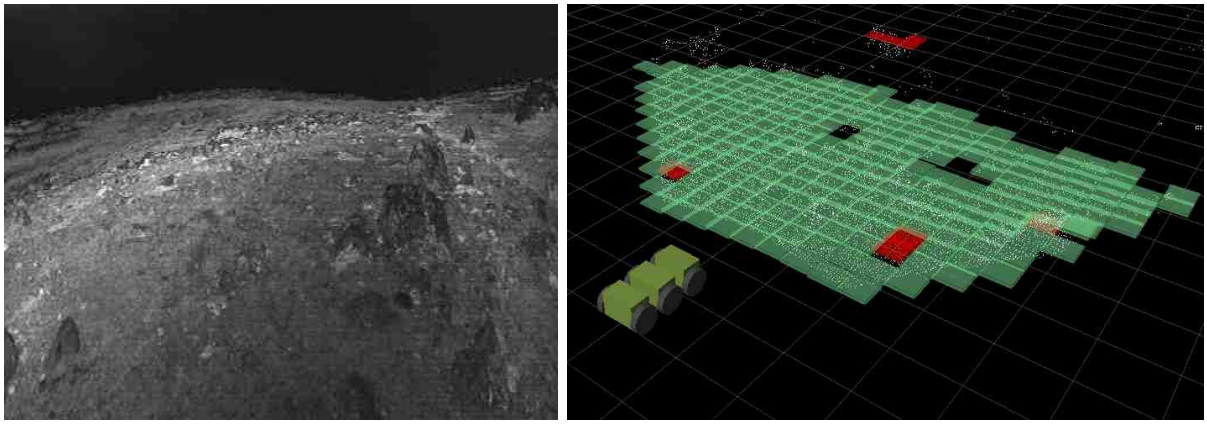

**Figure 5.11:** *Example intensity image (left). Each pixel of the intensity image also had a corresponding range. Example assessment from the Autonosys-lidar point cloud (right). The point cloud had many more points than in the stereo camera system, allowing for more of the terrain to be assessed.*

problem, a lighting-invariant system was created that makes use of a high-framerate lidar [103, 111]. Field test results from this system are presented next.

The high-framerate lidar used in these field tests was made by Autonosys Inc. The robot, with the Autonosys lidar, can be seen in Figure 5.10. Again, a differential GPS was mounted on the top of the sensor, but these data were only used for the ground-truth localization. The robuROC6 platform was again used, but with the different sensor configuration. The framerate of the lidar was approximately 2 Hz, and therefore the maximum speed of the robot was reduced to 0.5 m/s.

**Table 5.5:** *Results of a Autonosys-equipped mobile robot on a network of reusable paths attempting to seek a distant waypoint and return to the start. Most waypoints were reached (14 of 16) and the robot returned to the start in all tests. The columns are as in Table 5.1.*

| Test | Range to waypoint (m) | Estimated final range to waypoint (m) | Actual final range to waypoint (m) | Final localization error (m) | Distance to waypoint from start on network (m) | Total network length (m) | Total distance traveled (m) | Reached waypoint? | Returned to start? |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 50.0 | 0.6 | 1.2 | 1.7 | 54.7 | 65.1 | 160.0 | Yes | Yes |
| 2 | 44.7 | 0.5 | 3.3 | 3.2 | 50.5 | 64.1 | 165.9 | Yes | Yes |
| 3 | 45.0 | 0.3 | 0.4 | 0.7 | 46.7 | 63.1 | 152.9 | Yes | Yes |
| 4 | 45.0 | 23.0 | 23.2 | 0.2 | 45.1 | 59.2 | 138.0 | No | Yes |
| 5 | 50.0 | 7.3 | 6.0 | 1.3 | 57.3 | 138.8 | 411.9 | Yes | Yes |
| 6 | 50.0 | 0.2 | 2.7 | 2.7 | 52.4 | 141.3 | 395.0 | Yes | Yes |
| 7 | 45.0 | 1.1 | 2.4 | 2.3 | 45.9 | 51.6 | 123.3 | Yes | Yes |
| 8 | 45.0 | 0.1 | 2.0 | 2.0 | 45.9 | 60.4 | 147.0 | Yes | Yes |
| 9 | 45.0 | 1.7 | 1.8 | 0.2 | 61.2 | 76.0 | 182.4 | Yes | Yes |
| 10 | 45.0 | 2.2 | 0.9 | 1.4 | 65.6 | 111.8 | 274.0 | Yes | Yes |
| 11 | 45.0 | 0.7 | 0.8 | 1.5 | 47.9 | 52.3 | 129.3 | Yes | Yes |
| 12 | 45.0 | 7.5 | 8.0 | 1.6 | 46.6 | 99.4 | 337.5 | Yes | Yes |
| 13 | 50.0 | 1.2 | 1.6 | 2.2 | 53.3 | 69.4 | 162.0 | Yes | Yes |
| 14 | 50.0 | 1.9 | 3.8 | 3.7 | 51.7 | 56.4 | 133.7 | Yes | Yes |
| 15 | 50.0 | 1.7 | 1.9 | 3.6 | 58.4 | 76.2 | 200.5 | Yes | Yes |
| 16 | 50.0 | 20.2 | 21.5 | 1.4 | 64.1 | 82.7 | 198.5 | No | Yes |
| Total | 754.7 | 70.2 | 81.6 | 29.8 | 847.3 | 1,267.8 | 3,312.0 | 14 of 16 | 16 of 16 |
| Mean | 47.2 | 4.4 | 5.1 | 1.9 | 53.0 | 79.2 | 207.0 | 0.875 | 1.000 |
| Total when reached goal | 659.7 | 27.0 | 36.9 | 28.2 | 738.1 | 1,125.9 | 2,975.5 | 14 of 14 | 14 of 14 |
| Mean when reached goal | 47.1 | 1.9 | 2.6 | 2.0 | 52.7 | 80.4 | 212.5 | 1.000 | 1.000 |

An example intensity image (used for visual landmark detection and matching [103]) is shown in the left of Figure 5.11. The point cloud used by the Point Cloud safety layer was much more dense than in the stereo camera tests. The points were subsampled based on volume, and filtered to remove outliers. The filtering removed incorrect points that would arise when the laser spot measured the edge of an object and averaged the range with the background range. The assessment, shown in the right of Figure 5.11, detected locally flat terrain out to 10 m, and locally vertical terrain out to 20 m, with a higher

**Table 5.6:** *Manual interventions in the tests using the lidar-equipped robot. In the* reposition *interventions, an operator sent commands to the robot for the purpose stated in the description of the intervention. In the* obstacle *interventions, an operator manually added an obstacle at the robot's position. In* traction *interventions the operator did something to help the wheel traction.*

| Test number | Intervention type | Description of intervention |
|---|---|---|
| 4 | Obstacle | An obstacle was manually added to prevent the robot from driving into a pool of water. |
| 8 | Traction | Pushed down on a wheel to aid in wheel traction over a wet, mossy rock while repeating a path with a tight turn. |
| 10 | Obstacle, Traction | Three manual obstacles were added to prevent the robot from attempting to drive across a small stream. While repeating a path up a wet hill, the operator pressed down on the rear segment of the robot to improve the wheel traction. |
| 11 | Obstacle | A manual obstacle was added when the robot was experiencing wheel-slip in a wet, mossy area. Once the obstacle was added the robot backed out and continued the test without further intervention. |
| 12 | Reposition, Traction | The robot was returning along a path that made a turn while going up a slope. However, the robot was unable to reverse the maneuver on the first attempt. It drifted from the path and did not match against the map for just over 1 m (the distance threshold for these tests), so it stopped. The operator returned the robot to a previous point on the path and let it attempt the difficult section again, this time while pressing down on the front segment of the robot to provide more wheel traction. |
| 15 | Traction | Pushed down on a wheel to aid in wheel traction over a wet, mossy rock while repeating a path up a hill. |

vertical threshold used for the region beyond 10 m. This allowed obvious obstacles to be detected from a significant distance.

### 5.2.1 Results from Lidar-Based NRP

The results from the tests of lidar-based NRP are in Table 5.5. The robot successfully reached 14 of the 16 distant waypoints and it successfully returned to the start in all the tests. It drove a total of 3.31 km, creating a cumulative network length of 1.27 km.

Figure 5.12 shows an overview of test 5. This test was done in complete darkness, emphasizing the lighting invariance of the Autonosys-based visual-teach-and-repeat technique. The rover was asked to seek a distant waypoint that was 50 m straight ahead. As before, the robot had no prior knowledge of the terrain. It successfully avoided many obstacles and arrived within the waypoint capture threshold (8 m for all tests). The robot

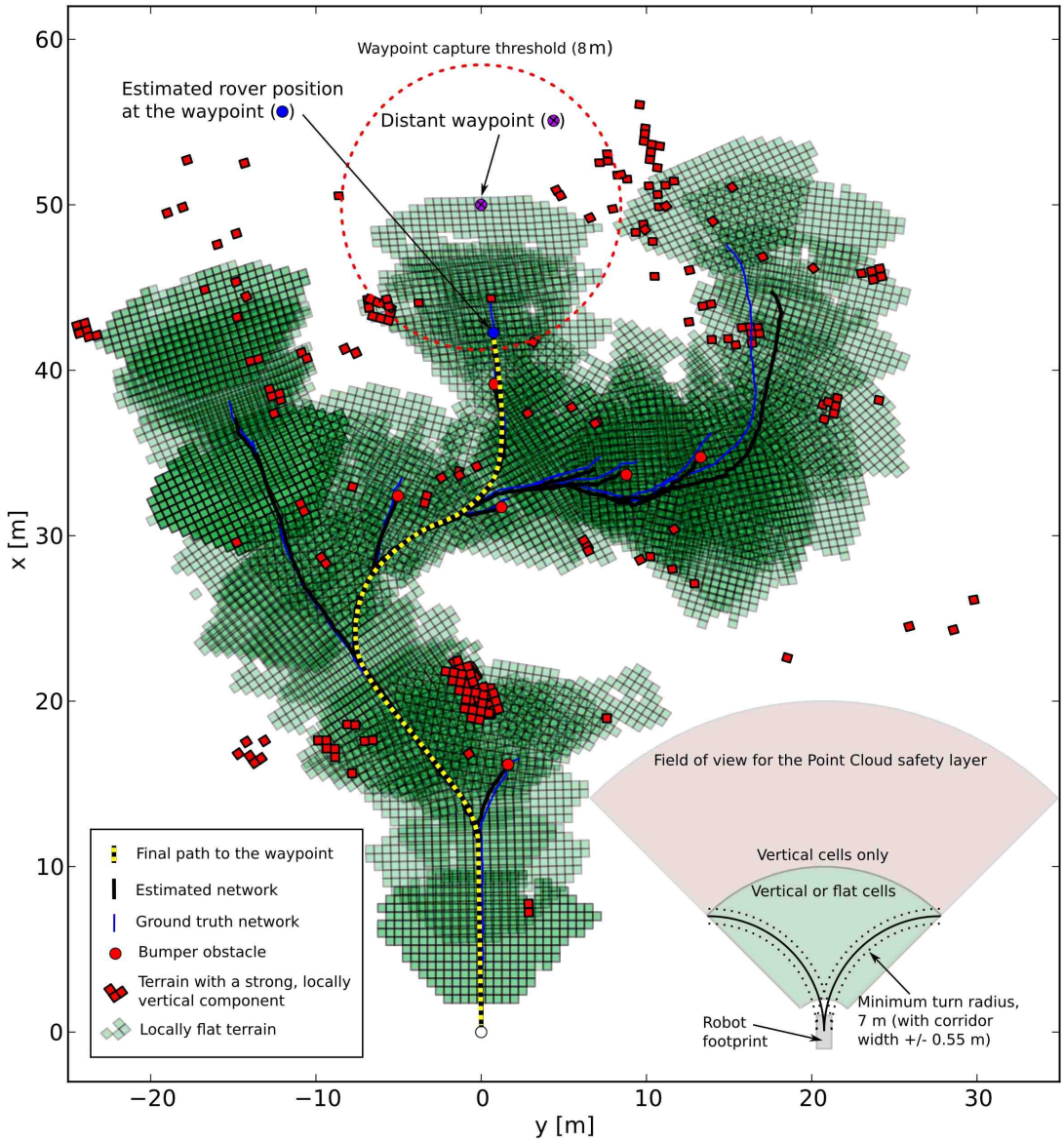


**Figure 5.12:** *An overview of test 5 when using the lidar-equipped robot to navigate to a distant waypoint and return to the start. Note how much more terrain data are available to the planner than in the case of the stereo-camera system. This is due to the increased point cloud range (see the legend in the bottom right). The area was strewn with rocks (the scattered red cells) that were not traversable by the robot. Also note that this test was done at night, in total darkness.*

then returned to the start position. The Autonosys lidar allowed for much more terrain data to be used in the Point Cloud safety layer and when planning new paths. However, the localization estimate was much more uncertain than the stereo-camera system, meaning that seeking more distant goals in a single command cycle was troublesome. Further work, including motion compensation, is underway to address this issue [127, 128, 129].

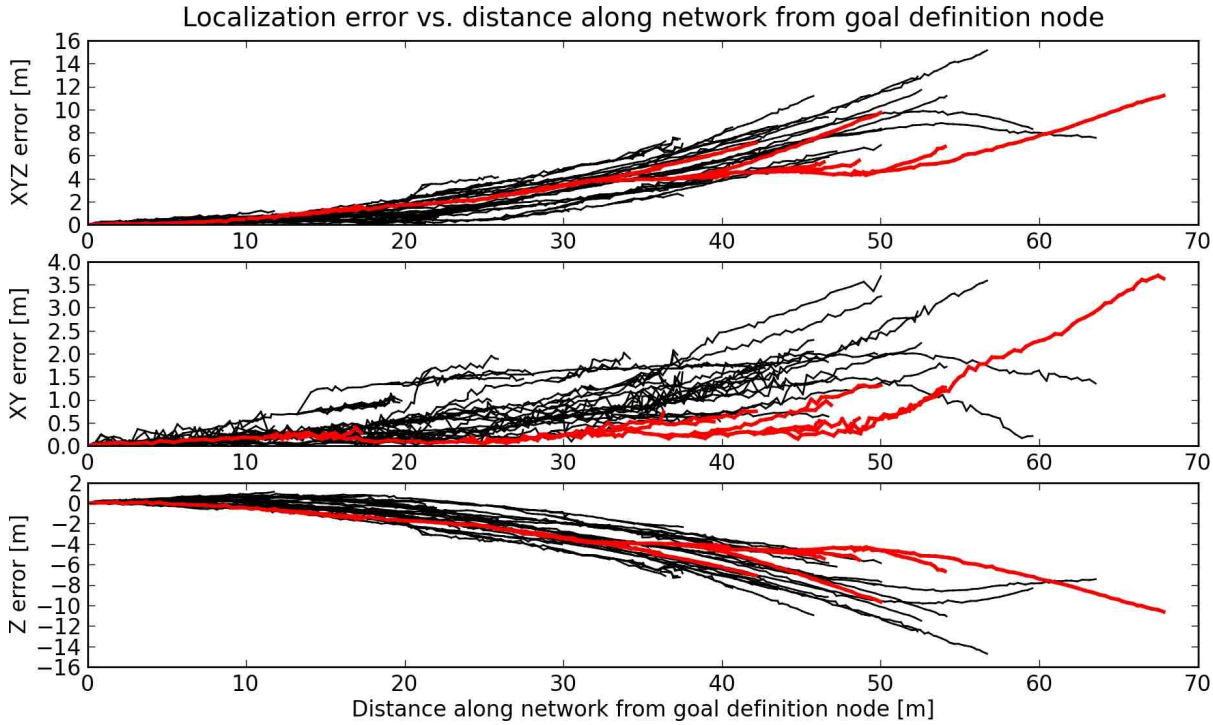


**Figure 5.13:** *Localization error versus distance along the network for the tests using the Autonosys-equipped robot. Note the network structure of the individual test that is highlighted in red. This is from test 5.*

In these tests, manual interventions (listed in Table 5.6) were primarily necessitated due to the traction performance of the vehicle, or because there was a lot of standing water that was deemed an unnecessary risk for the robot (it was unclear if it would always stop in time). Again, the water hazard might be addressed by adding an additional safety layer to the system. The traction problem would typically mean the robot stopped moving along the path and it would spin a single wheel. This was a low-level hardware issue with the vehicle that could not be changed (ideally, power would be routed to the wheels that were not slipping). Instead, we pushed down on the wheel that was spinning, thereby allowing it to generate more traction. This could be thought of as a vehicle design issue, but it also indicates that sometimes a path is not drivable in both directions (this breaks our second assumption about the robot). At least two options are then available: either make the safety layers more strict to reduce the frequency of this error, or make use of single-direction paths. These are considerations of later works.

The growth of the localization error versus distance traveled is plotted in Figure 5.13,

similar to the stereo-camera plots in Figure 5.9. Test 5 is highlighted in red and the network structure is visible. Again, we see a trend of superlinear error growth with distance traveled, but this time over much shorter distances. Similarly to the stereo camera system, we see a significant vertical drift (bottom axes of Figure 5.13). The high-frequency noise on the error is due to the back and forth continuous scanning of the sensor. As mentioned, no attempt was made to compensate for this motion at this stage. Instead, the camera (operating at about 2 Hz) was modeled as capturing the whole frame simultaneously. This is a major simplification, but it is interesting to see that the system still worked. Again, one can speculate on what the error would grow to if the robot were dead-reckoning all the time and never using a network of reusable paths. It would still have to back up out of dead ends, and back up from nearby obstacles, and the localization error would drift during those maneuvers.

## 5.3 Discussion

This section presents a discussion on the performance achieved using NRP. First, we identify the challenges in robust, autonomous exploration (Section 5.3.1), next, we show how NRP can be thought of as a physical embodiment of an RRT (Section 5.3.2). Finally, we attempt to present an intuitive explanation for the improved performance of NRP when compared to other approaches to GN&C (Section 5.3.3).

### 5.3.1 Challenges in Robust, Autonomous Exploration

Even though we take localization error as the primary criterion for measuring success, we can also look at the other challenges that are faced in autonomous traverses by a mobile robot. Many of these challenges are avoided in the NRP approach.

Map merging, for example, can result in artifacts that are exacerbated by localization error. Yet, most systems require maps to be merged in order to express the terrain traversability in a single frame for planning. Using NRP, we have thus far avoided any map merging, instead using many topologically close (i.e., nearby along the network) local-assessment maps during planning. A local assessment map is created from data

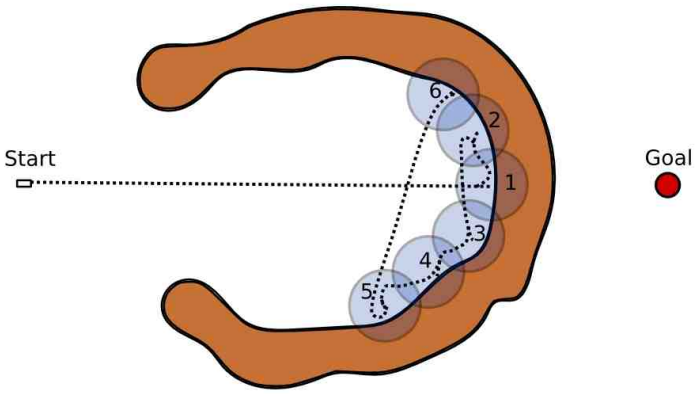


**Figure 5.14:** *Challenges in map merging include the potential for getting stuck if the map does not capture the scale of the obstacles. In this example, if map data are forgotten as new data are added (i.e., the robot only keeps the data from the last six scans), the robot will be stuck going back and forth on the inside of the obstacle and the robot will not reach the goal.*

at a single sensor pose. This has avoided many of the problems encountered when local maps are incrementally added to the current, global map. For example, when adding new assessment data to a global map the robot can encounter false steps when the elevation is not correct, real openings can be closed off in the map when the merging overlaps the maps by too much, or false openings can appear when merging does not have enough overlap between maps. The robot will typically forget distant or old data to avoid these problems from building up over long distances. However, this can lead to the robot getting trapped when it cannot build a traversability map large enough to capture the obstacle that needs to be avoided. As in Figure 5.14, consider the case where the robot is trying to get past a large semicircular obstruction; if it does not build a large enough map it may become stuck in an endless cycle going back and forth on the inside of the curve because it keeps forgetting about what it has already seen (also, consider what is happening to the localization estimate while the robot attempts this traverse). As was shown, these problems are avoided by using the NRP approach.

Another challenge arises when considering the problem of point turns. Not all vehicles are capable of such a maneuver (the robot used in our field trials was not able to turn on the spot). Even those robots that can turn on the spot can be operating in terrain where such a maneuver is impossible (stability concerns, or a tight area with a robot that is longer than it is wide). Once in these tight spots, it can be difficult to find a path that

can back the vehicle out, and when a robot does not have complete sensor coverage (and many common robots fall into this category), it can become unsafe to reverse blindly into terrain that is believed safe. However, using NRP allows a robot to reverse along the inbound path, even when it cannot see the terrain into which it is driving (in reverse).

### 5.3.2 A Physical Embodiment of an RRT

Next, we show that the proposed system (with some conditions) is a physical embodiment of an RRT, and that it could be made to be a physical embodiment of any planning algorithm that incrementally builds a spanning tree rooted at the goal definition node. What we mean here is that the world is serving as its own map, and the planner is operating in the world, not a virtual map; this is enabled by the ability to repeat paths using VT&R.

We wish to show that the proposed NRP system is a physical embodiment of an RRT so that we can claim the same properties of an RRT. One such property is that an RRT is probabilistically complete, meaning that when a path exists, the probability of finding that path goes to 1 as the number of iterations goes to infinity [51].

Let us consider a simple and ideal NRP system. The only method of terrain assessment is to actually attempt to drive a piece of terrain, but this assessment is error-free (a perfect bumper, and let us use 'bumper' with a little latitude on the meaning). Additionally, because an RRT planner assumes perfect localization, we, too, use that assumption; as in the assumptions about the robot, we assume a perfect visual-teach-and-repeat system.

**Theorem 5.1.** *In the absence of localization error, a ideal NRP system using a perfect bumper for terrain assessment is a physical embodiment of an RRT (i.e., where the world is its own map).*

*Proof.* When using NRP, all steps in Algorithm 2 (from Section 4.3.5) are either identical to the case of virtual planning, or the steps have physical implementations. Thus, NRP is a physical embodiment of an RRT. Consider line 1, the network is a graph, thus all these elements make up the actual network. Lines 2 and 3 have no change in implementation.

In line 4 the robot not only calculates the nearest node on the tree, it also travels to that node (recall the robot can travel anywhere on the network). Lines $5-7$ are carried out simultaneously as the Steer() function is implemented by actually having the robot attempt to drive from $x_{\text{nearest}}$, and the ObstacleFree() function is evaluated by the terrain assessment while the robot drives into new terrain. The network is grown any time the robot is driving into new terrain, therefore the traversable part of the new segment is added to the network. □

There is a further generalization of the notion of physical embodiment. Consider the NRP system we used in our real-world tests; it created and then operated on a tree of paths (i.e., no loop closure, no cycles, no adding a second inbound edge to an already existing node). In fact, this system was maintaining a spanning tree of all the nodes on the graph, and we can therefore extend the set of planners that can be physically embodied to include any planning algorithm that incrementally builds a spanning tree from the point where the goal is defined.

An RRT is one example, but one could also consider a system that uses a breadth-first or depth-first search, if a suitable graph representation of the state space were given (e.g., an eight-connected graph, a state lattice [52]). Note that the practical requirement that the network be a single tree (i.e., no cycles), containing the goal-definition node (this can be thought of as the start) rather than the goal itself, precludes the use of certain algorithms such as RRT-Connect [51]. This requirement is to avoid the problem of expressing all of the assessment data in a single global coordinate system.

Unfortunately, none of the spanning-tree planning algorithms are optimal with respect to path cost [61]. This raises the question: what is required to allow for optimal planning on a network of reusable paths? This line of questioning is the subject of future work and the preliminary discussions are in Section 5.4.

### 5.3.3 Breaking a Vicious Cycle

We can look at the interactions between localization, mapping, and planning, to gain insight into the improved performance experienced when using a network of reusable paths, compared to a system using a classic approach to GN&C. The top of Figure 5.15

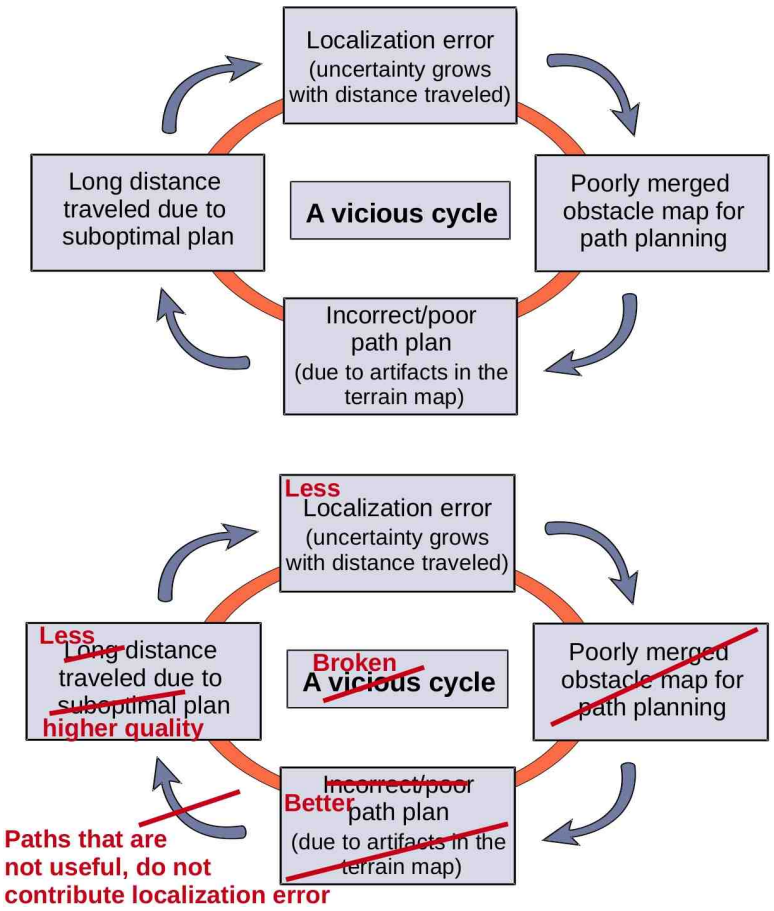


**Figure 5.15:** *A vicious cycle arises in the classic approach to GN&C (top). Using a network of reusable paths breaks that cycle (bottom). In classic GN&C the localization uncertainty grows with each movement, the localization is used to create a global map of the recently observed obstacles so localization errors lead to poor quality maps. Poor quality maps lead to poor plans which then lead to longer distances being traveled, and again, greater localization uncertainty. A system using a network of reusable paths experiences less localization error, as error is accumulated along the network back to the goal definition node. No global map of assessments is made, thus avoiding the map merging problems. This leads to better path plans and less distance traveled. Finally the cycle is broken because paths that are not useful do not contribute to the final localization error. The robot rolls back error as it reverses along the network.*

attempts to capture the detrimental interactions between these elements in a classic approach to GN&C. We can think of it as a vicious cycle. Localization error tends to grow with distance traveled, then this larger localization error introduces more inconsistencies into the single-frame map containing all (or just the recent) terrain assessments. This poor map might include phantom obstacles (false positives) and non-existent openings (false negatives). When this map is used for planning, the result will be a poor path plan. Then, because the poor plan will either be longer than necessary, or impossible

and thus require a new plan, the robot will end up driving farther. This results in even more accumulated localization error, and the cycle reinforces itself.

A system using a network of reusable paths breaks this vicious cycle. Less localization error is accumulated, and no attempt is made to make a single frame holding multiple local terrain assessment maps. All potential segments are instead transformed into the frame of the local assessment and the problems associated with map-merging are avoided. The result is a higher quality path plan because there is a reduction in the number of map errors, therefore, the vehicle travels a shorter distance. Additionally, the cycle is actually broken because the final localization error is only accumulated along the final path to the goal from the goal definition node on the network. This means that paths that are not useful do not contribute to the final localization error (e.g., dead ends have no lasting impact).

## 5.4 Future Work

One focus of our future efforts will be to incorporate other planning paradigms into the network of reusable paths approach. For example, two additional capabilities are required in order to embody some types of optimal path-planning algorithms: (i) the cost of an edge must be considered, and (ii) the system must be able to update the edge connections between states when a lower-cost path is found.

The edge cost is easily added by measuring the experienced cost (e.g., time, distance, power, growth of localization uncertainty), and it is interesting to note that this is essentially the actual edge-cost, rather than the predicted edge-cost used in other map representations and path planners.

Updating edge connections requires the visual-teach-and-repeat system to be able to do loop closure, and this is one of the next logical abilities to add to the NRP system. This would eliminate the restriction that the network is a tree (which was the case in our tests so far, even though the algorithm works on arbitrary networks as well). We could, for example, still create a minimum spanning tree of the network if we have some restrictions on what constitutes a valid edge cost (non-negative scalar), and thus use

Algorithm 3 (from Section 4.3.5). It would also allow for the use of algorithms that can be thought of as constructing a minimum spanning tree, or a connected graph of nodes where the edges can be updated. These algorithms include: Dijkstra's algorithm [120], A* [49], RRG [53], and RRT* [53]. If the network of reusable paths were built upon one of these optimal planning algorithms, then the final path to the goal would be optimal (in the same way that the planner is optimal). Note that the requirement that it be a connected graph containing the goal definition node precludes D*-like algorithms, as those algorithms build and maintain a spanning tree from the goal itself, which is not always part of the existing network.

There are two other motion planning approaches that should be mentioned as they appear similar to RRTs; these are planning using generalized Voronoi diagrams [130], and planning using a Probabilistic Roadmap (PRM) [131]. Unlike the RRT approaches, the Voronoi-based approaches do not explicitly account for the limited vehicle kinematics (such as a minimum turning radius). A PRM can incorporate vehicle kinematics; however, neither of these approaches incrementally construct the graph from a single point, making them unsuitable for embodiment using this method and it also unclear how they could plan to reuse paths in the same manner as the RRT.

The ability to repeat a previous path is fundamental to a physical embodiment of the above path planners. It is interesting to note that by creating a physical embodiment of the planning algorithm, the planner is able to use the actual costs to find the lowest-cost path to the goal. It also allows the planner to use the actual vehicle kinematics

In each of our tests, the robot successfully returned to the start, but this is unlikely to always be the case. Later systems may need the ability to do path repair. This ability seems to share many similarities with the ability to do loop closure. Another option would be to monitor the teaching system and avoid driving into areas where the robot would be unable to later localize. We have done preliminary testing of a safety layer that monitored the number of local landmarks that were being added to the map when driving into new terrain. It added obstacles to the map when the landmark count was too low. Mitigating localization failures is an active area of development. Further challenges, and our approach to meeting them, are discussed by Barfoot et al. [111].

## 5.5 Conclusions

In the last two chapters, we have extended visual teach and repeat to a network of reusable paths. Using NRP for seeking distant waypoints has advantages over other approaches. These advantages include reduced localization error and greater robustness. From a localization perspective, NRP allows the path planner to plan based on how the robot has done.

This simple SLAM system achieves many of the benefits of a single-coordinate-frame SLAM system for the goal seeking problem, but at relatively low computational cost. Localization error and uncertainty only accumulate along the best path found to goal, even without a priori knowledge of the terrain. The result is that the system behaves as a physical embodiment of a rapidly-exploring random tree.

In this chapter, we presented two sets of field trials carried out at planetary analogue sites. The vehicle traveled a total of more than 14.4 km in these two tests. The first test used a robot equipped with a stereo camera, and the second was a lighting-invariant system where the robot was equipped with a high-framerate lidar. In these tests we saw that the two assumptions (see Section 4.3.1) that were made, being able to stop before obstacles and always being able to repeat previous paths, are generally reasonable. Much of the future work will be focused on how to deal with the breakdown of the second assumption, (e.g., what happens when the appearance or the traversability of a path changes, and what happens if a path is not reversible).

These experiences have yielded insights into both what works, and what needs improvement in the next iteration. Using a network of reusable paths is a promising approach to navigation and planning for a mobile robot, and we expect to continue developing the concept in the years to come.

# Chapter 6

# Exploration using a Network of Reusable Paths

The Mars Exploration Rovers (MERs) have driven over 42 km, visiting many sites of scientific interest along the way. The exploration strategy for each rover was serial in the sense that scientific objectives were completed at one site before departing for the next. This means the robot remained in place while mission controllers decided which measurements to collect [68, 69]. Figure 6.1 shows the traverse map for the Spirit MER as of sol 2555. Note that many of the sites that were visited were near each other, and on several occasions the rover would roughly follow its previous track to return near to a previous position. Returning along a previous route could take several sols and many command cycles.

The coming decades will see sample-return missions to both Mars and the Moon. Here, we advocate for a planetary exploration strategy that allows sites of interest to be studied in parallel, rather than in series. We believe this better supports the overarching aims of sample-return missions, as a methodical down-selection process may be efficiently employed to identify the key specimens to be returned to Earth. We show that by using a *Network of Reusable Paths* (NRP) [59, 60] a rover can revisit places of scientific interest and thus allow the study of sites in parallel [62, 63]. This new approach was field tested in a mock Lunar sample-return mission conducted near the Sudbury impact crater in

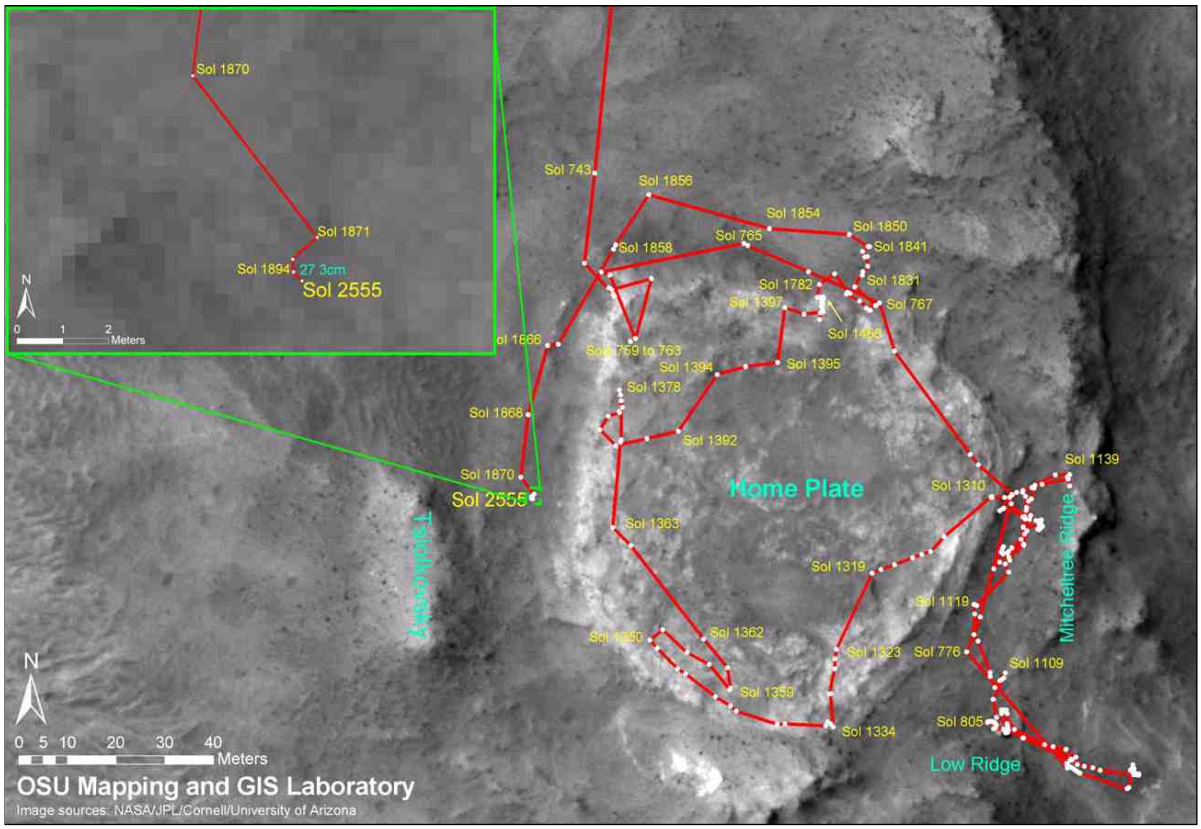


**Figure 6.1:** *Traverse map at sol 2555 for the Mars Exploration Rover, Spirit. Credit: OSU Mapping and GIS Laboratory, NASA/JPL/Cornell/University of Arizona.*

Canada[1], and in a mock mission in the Mistastin impact structure in Northern Labrador, Canada, where the robot was used as an astronaut assistant [64, 65, 66]. The robot drove a total of 3.9 km and 8.2 km in these tests, respectively. The test locations can be seen in Figure 1.4. In this chapter, we emphasize three points from our previous work [62, 63]:

1. *NRP allows a robot to return to a previously visited position with a single command,*
2. *this allows for parallel exploration of sites of scientific interest, and*
3. *parallel exploration allows for an efficient down-selection process to identify key samples for return.*

We discuss how NRP can be used in planetary surface exploration in Section 6.1, and in Section 6.2 we present the mock sample-return mission. Section 6.3 is a brief overview of the astronaut-assistant scenario. In Section 6.4, we identify other uses of NRP, and in Section 6.5 we identify challenges to be expected when doing planetary exploration using NRP, along with some future works.

[1]A video showing an overview of this mission is available at: http://youtu.be/kJQdo6guglE

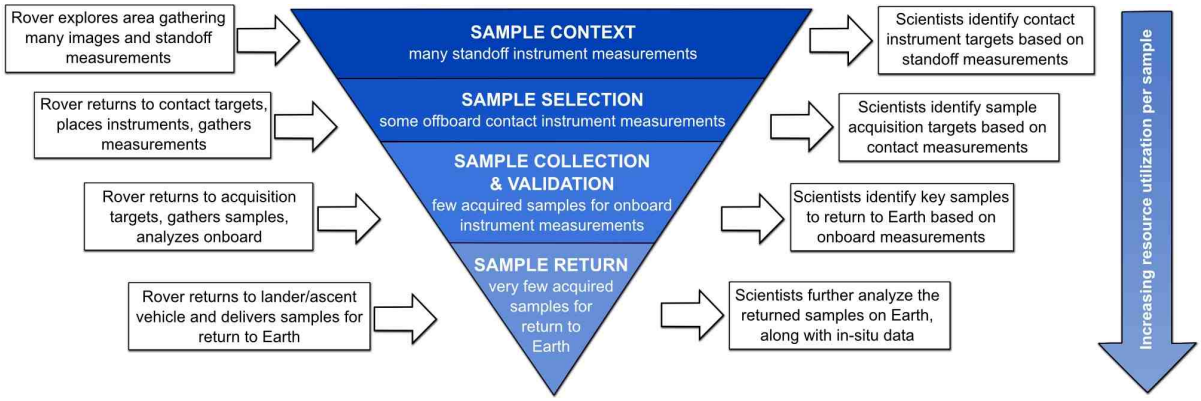


**Figure 6.2:** *A methodical down-selection process is enabled by using a network of reusable paths. There are a decreasing number of samples at lower levels to accommodate the higher resource usage per sample.*

## 6.1 A Network of Reusable Paths for Planetary Surface Exploration

In the sample-return scenario, NRP allows for a methodical down-selection process as shown in Figure 6.2. This process is possible because the robot can return to any previous position, and therefore tasks and waypoints can be defined relative to any previous position. In a sense, this allows the instructions to be parallel, in that in a sequence of complex steps, each step is not defined relative to the predicted end-point of the previous step. Instead, NRP encourages setting short-range, parallel objectives that can be reliably completed in a single command cycle and then built upon in later command cycles once mission controllers have reviewed the resulting telemetry.

Consider the example network in Figure 6.3. Here, there are three sites of interest that are being investigated in parallel. While operators on Earth discuss a decision on where to sample at site A, they can send the robot to site B and then C to collect imagery before returning to site A. Then, while the robot is sampling at site A the mission team can use the data from Site B and C to select another potential sampling site. The rover does not need to loiter at a particular site of interest until all the work there is done. It is able to leave and return.

In this way, the mission team can use a methodical approach to selecting the most promising samples to return to Earth. As in Figure 6.2, a great deal of data about the

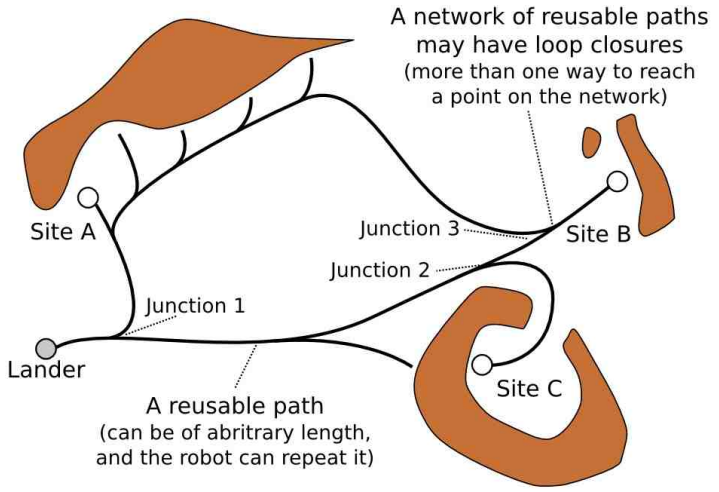


**Figure 6.3:** *A simple network of reusable paths is shown in black. The robot can return to any point on the network and can grow the network into new areas. To go from site B to C, the rover reuses the previous paths by traveling through junction 3 and then 2, before going to site C.*

area are collected, at many sites in parallel, using imagery and standoff measurements. Scientists then use these data to identify targets for contact measurements. The robot returns to the selected sites and carries out the contact measurement tasks. Scientists use the results of the contact measurements to select the best candidates for sampling. Again, the robot returns to previous points, this time to collect samples. Once the samples are collected the scientists can determine the key samples to be returned to Earth. The rover can return the selected samples to the lander/ascent vehicle on the network with one command.

Many variations on these ideas are possible, and more of these options are discussed in Section 6.3 and 6.4. In the above example the benefit is that mission operators have the flexibility to efficiently delay sampling decisions pending a more thorough investigation of the data already on Earth. This can dramatically improve the efficiency of the system. In practice we found it to be as though there were multiple rovers with offset command cycles.

## 6.2 A mock Lunar sample-return mission using NRP

The mock Lunar sample-return mission was conducted near the impact crater in Sudbury, Ontario, Canada. This was one of three missions [64] we conducted that were funded by the Canadian Space Agency. In this section, we give a brief overview of the mission, discuss the robot configurations that were used, and then present details of the mission time line, with an emphasis on the parallel exploration made possible by NRP.

### 6.2.1 Mission Overview

The field test was a robotic Lunar analogue mission in support of future sample-return missions to the Moon (and Mars), with a target of the South Pole Aitken basin. The primary objectives of the mission were to perform: (i) an in situ investigation of geology in a Lunar analogue environment, and (ii) an investigation of the formation processes and resource potential of impact crater(s).

The mission scenario lasted two weeks, with two different rover configurations (see Figure 6.4). For an overview of the mission operations, see Moores et al. [66]. Command cycles were nominally two hours in length and communication was only available during a window at the beginning and end of the cycle. Instructions were sent to the robot at the beginning of the cycle and telemetry was sent back at the end. This meant that there was very little time to review the results from the previous command cycle before the instructions for the next were sent.

In the first week, 24 command cycles were carried out, creating a network with 0.23 km of paths while driving a total of 1.0 km. The second week had 19 command cycles, a 0.44 km network and 2.92 km of total driving (3.9 km in total in the two weeks). Of the 17 samples that were collected and returned to the lander, ten were selected as the sample retention set (i.e., the samples that would have been returned to Earth for analysis). An overview at the end of the second week is shown in Figure 6.6.

**Figure 6.4:** *The robot as it was in the week-one configuration (left) and the week-two configuration (right).*

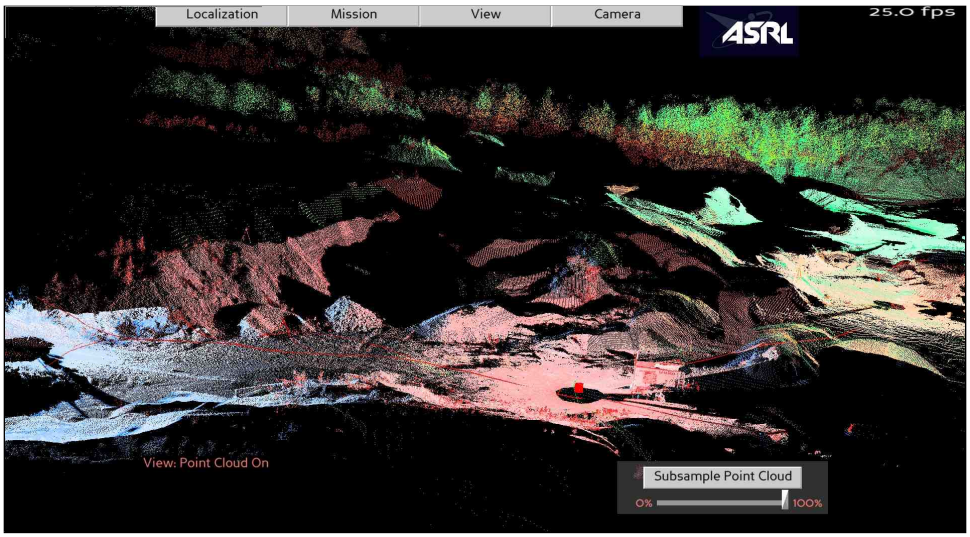


**Figure 6.5:** *The ground station software was used by mission control to plan rover paths and to check rover telemetry. It allowed operators to plan a path relative to any of the lidar scans, and it allowed the lidar scans to be aligned to give a higher quality model of the area.*

### 6.2.2 Rover Configurations

The mock mission used two different robot configurations, as shown in Figure 6.4. The week-one configuration is on the left, and the week-two configuration is on the right. In both, the robot used a stereo camera as the primary sensor for adding to, and repeating paths on, the network of reusable paths. The vehicle, a robuROC6 made by Robosoft SA,

had three passively articulated body segments. All the primary guidance, navigation, and control (GN&C) sensors and software were onboard the vehicle, but there were scientific sensors and tools that were not integrated with the vehicle.

In week one, the configuration favored onboard scientific capabilities over mobility. The vehicle had only simple GN&C capabilities. The stereo camera used for NRP was at the back of the vehicle, facing backwards. None of the other onboard sensors were integrated into the rover GN&C. The robot would turn and drive directly toward the current waypoint with only simple safety monitoring. If an obstacle was detected, the robot would stop and attempt to reach the next waypoint in the sequence of closely spaced waypoints. The CBRN Crime Scene Modeler (C2SM), made by MDA, was on the front of the robot. A ground-penetrating radar (GPR) was pulled behind the robot (and removed when the vehicle was repeating a path in reverse). Panoramic imagery was obtained using a DSLR camera on a GigaPan pan-tilt unit. Also available, but not onboard the robot, were a hand-held Raman spectrometer, an XRF spectrometer, and a drill used to obtain core samples. We had an Optech ILRIS-$3_6$D lidar that was located at the lander (see the first-week configuration in Figure 6.6). A GPS antenna was located near the stereo camera; however, GPS was only used to measure the ground-truth localization, and it was not available during the scenario.

In week two, the configuration favored mobility over sensor integration. The GPR and C2SM were not used, instead, the lidar was mounted on the front of the vehicle, and on top of that, a forward-facing stereo camera that was used for NRP. The onboard lidar allowed for many local scans to be taken. The rover GN&C was more sophisticated. The robot used the stereo camera, inclinometers in each segment of the vehicle, and wheel odometry to identify hazards. Onboard planning let the robot plan to avoid detected hazards. The planner was able to reuse the existing network, as well as plan paths into previously untraveled terrain [59, 60]. In these tests we used a differential GPS for the ground-truth localization, and again, the resulting data were not available to the robot or mission controllers during the tests.

In both weeks, mission control created traverse plans and reviewed the rover telemetry by using the ground station software (see Figure 6.5). In the second week, when the lidar

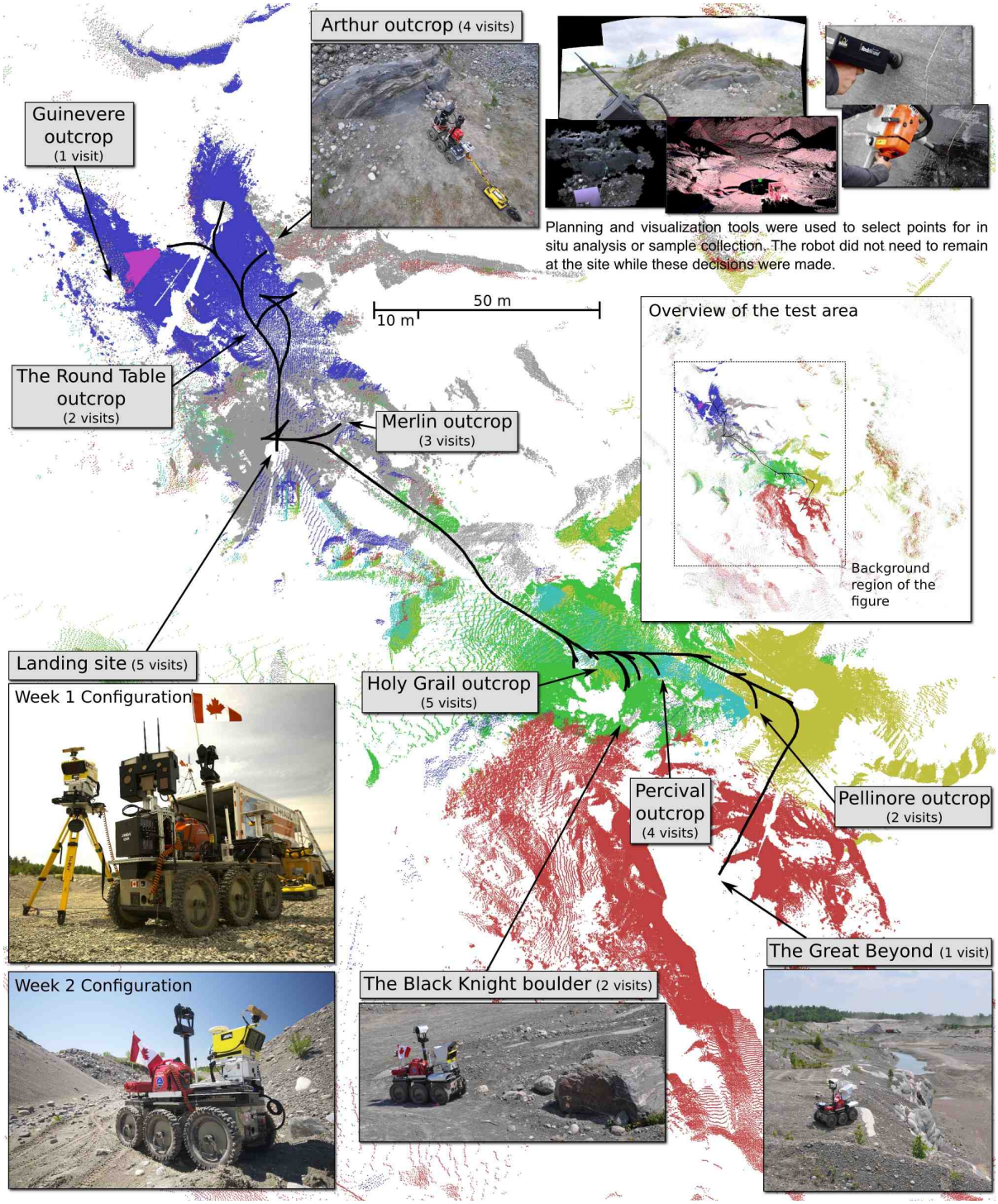


**Figure 6.6:** *An overview of the network of reusable paths (the black line) at the end of the sample-return mission. NRP allowed the robot to return to any position that was previously visited. This meant that mission control could delay analysis or sampling decisions at one site, and still continue to carry out operations at other sites. At the end of week two, the total length of the network of reusable paths was 440 m, and by then the robot had driven a total of 3 920 m.*

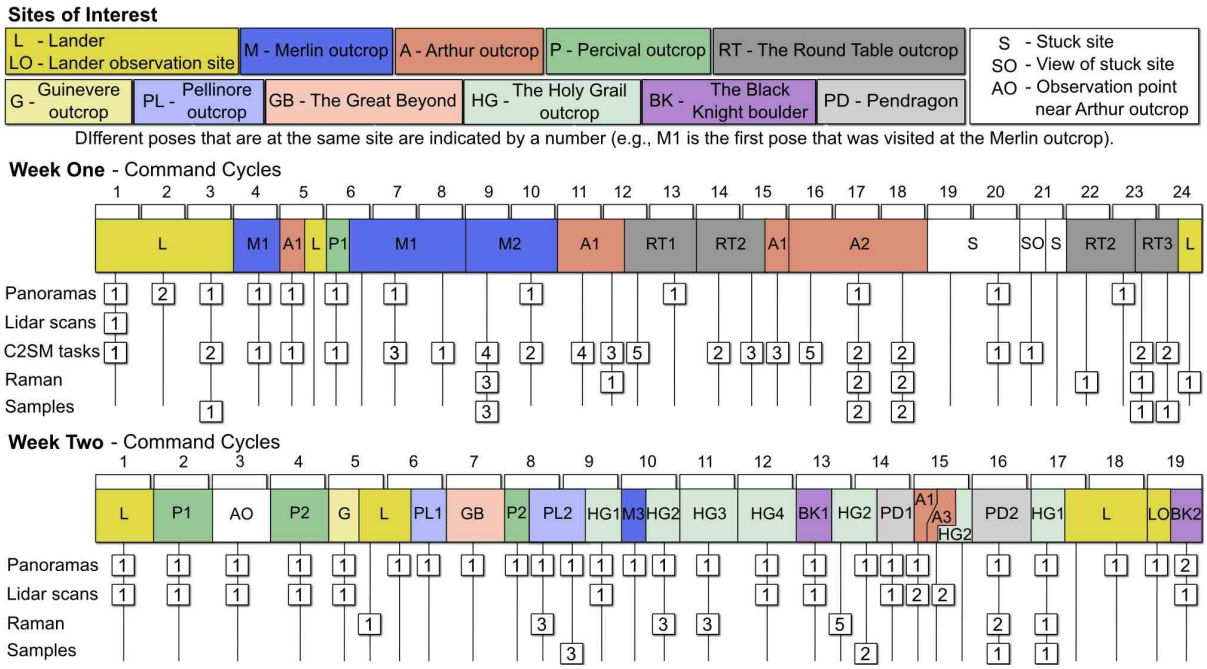


**Figure 6.7:** *An overview of the command cycles carried out in the first and second weeks of the mission. The type and quantity of the tasks that were done at each site are shown in the squares below the command cycle. A total of 24 command cycles were carried out in the first week, and 19 command cycles were carried out in the second week.*

was onboard the rover, the ground station could also be used to manually tie together multiple lidar scans rather than relying on the dead-reckoning localization from the visual odometry onboard the rover. Controllers could define waypoints relative to any point of the network. There was no privileged coordinate frame that all waypoints had to be set in, and the waypoints did not need to be from the robot's current position. Typically, waypoints were defined relative to a lidar scan.

### 6.2.3 Results from the Mock Mission

As this chapter is about the use of NRP, rather than this specific mock mission, we omit further details about the scientific sensors, sampling methods, and the resulting findings. Instead, we limit the presentation of results to those that are pertinent to this discussion, namely, traverses to and between sites of interest and, broadly, the tasks that were carried out at those sites, as those tasks fit into the down-selection process.

An overview of all the command cycles in the mission is shown in Figure 6.7. The

main sites of interest are color-coded at the top. Many of these sites had more than one distinct pose that was visited, these poses are distinguished by a unique number (i.e., M1 is a unique pose at the Merlin outcrop). In the overview we can see that in the first week, the rover spent the first three command cycles at the landing site. In the first command cycle, it took a panoramic image, a lidar scan, and observed the terrain using C2SM. The second cycle was used to collect two detailed panoramic images. The third cycle was used to collect another panoramic image, take two measurements using C2SM, and collect a sample of the material at the landing site. In the fourth cycle, the robot approached the Merlin outcrop and collected panoramic imagery and standoff measurements. At this point we begin to see the pattern of parallel exploration. In the fifth and sixth cycle the robot visited other sites and collected imagery and standoff measurements, before returning to the Merlin outcrop. While the robot was there, mission control reviewed the data from Arthur and Percival and planned future tasks for when the robot returned to these sites.

In command cycle 19 of week one, the robot attempted to reach an observation point near the Arthur outcrop (later visited in cycle 3 of week two), but it became briefly stuck in soft soil. In cycle 20, the rover collected imagery to aid in determining what had happened. The next command cycle was to back up and observe the point where the rover became stuck, and then attempt a similar traverse, which again, the robot could not complete due to soft soil.

At the end of the first week we marked the rover wheel positions at the sites of interest and manually drove the robot to teach it a new network that reached those same physical locations. This was necessary as the significantly different camera location (now pointed forward rather than reverse) changed the appearance of the scene and the paths could not have been recognized. The two networks, and the shared sites of interest, are shown in Figure 6.8.

Figure 6.6 gives an overview of the mock mission at the end of week two. Multiple long-range lidar scans are displayed in different colors. The network of reusable paths is shown in black. Figure 6.9 takes a more detailed look at command cycles 2 through 5 in the second week. This is an example of parallel exploration. In the even-numbered

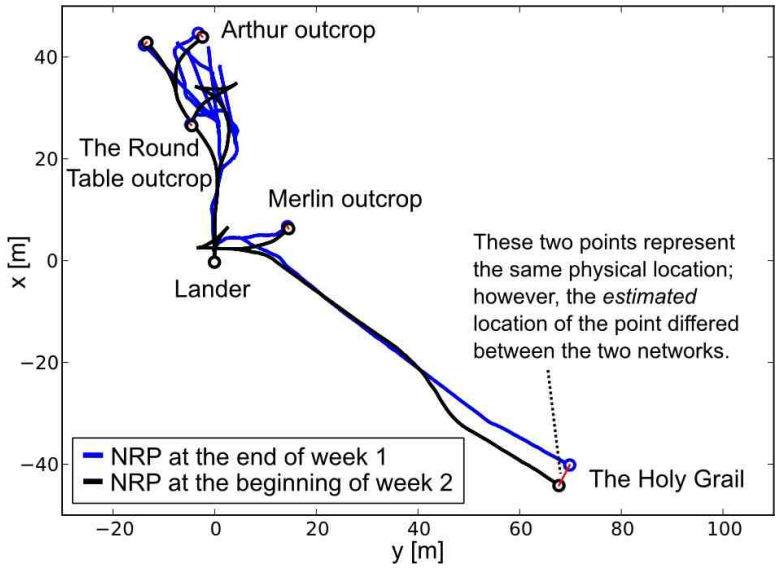


**Figure 6.8:** *The week-one network was manually retaught for the new camera configuration in week two. Key points of scientific interest were kept in the new network.*

cycles, the rover was exploring in the bottom right of the map. In the odd-numbered cycles the rover was exploring in the upper right. Mission controllers reviewed the even-cycle data during the odd cycles, and had instructions ready for the robot at the beginning of the next command cycle. In this case, it was as if there were two robots exploring two different areas. With a serial approach, it would take three cycles to explore each individual branch (one where the rover did not move while mission controllers reviewed the telemetry and waited for the next communication window), and in order to begin exploration of the second branch, the rover would need another two or three command cycles to return to nearby the lander. Thus, in a serial approach, the same exploration might be expected to take 8 or 9 command cycles, rather than 4 (100% to 125% more cycles to explore the same two areas).

We have attempted to determine how many command cycles the entire mission would have needed had a serial exploration strategy been used. We began by estimating how many command cycles the robot would take at each site. For example, we estimated that the Merlin outcrop would have taken $8-9$ command cycles to investigate to the same degree. This range comes from assuming the rover sits idle for one cycle while mission control reviews data and waits for the next communication window. So, one cycle as in week one, cycle 4, then an additional cycle idle, then $4.5-5$ cycles as in week one, cycles

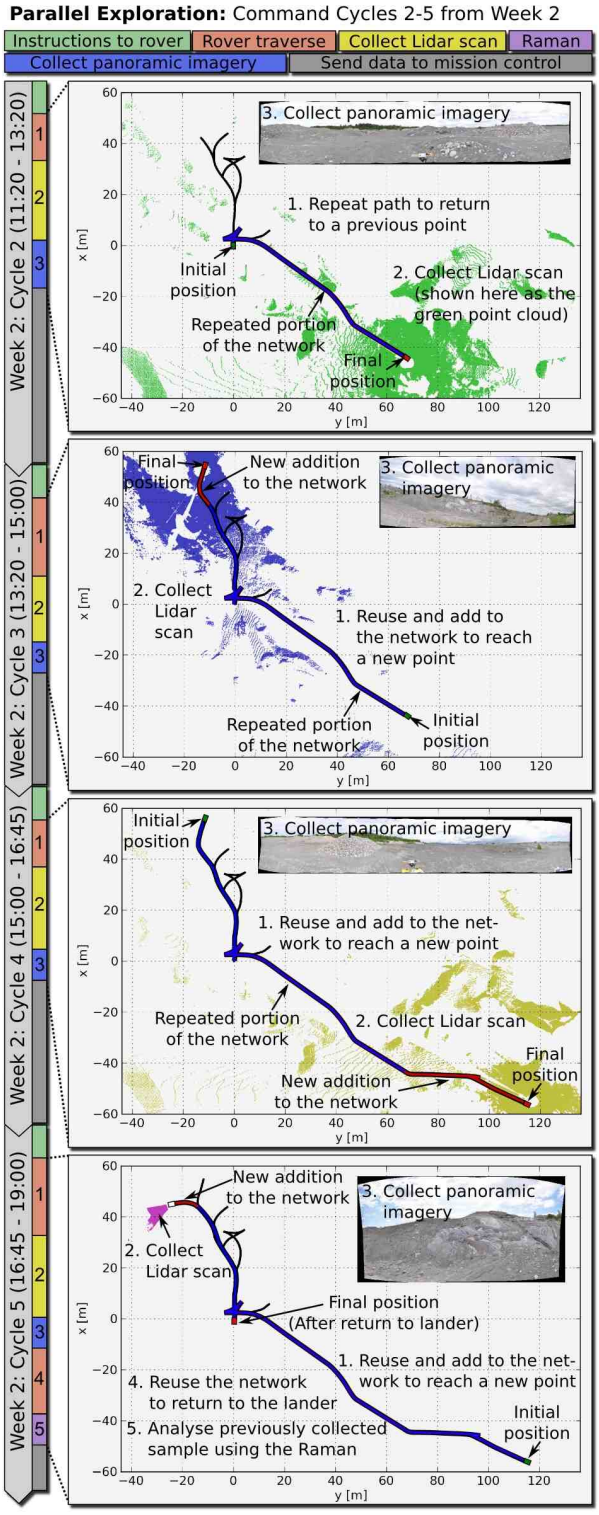


**Figure 6.9:** *Parallel exploration was done between the odd and the even command cycles. In cycle 2 of week two, the robot collected a lidar scan and panoramic imagery. In cycle 3 the robot was sent to another area to collect more data while mission control reviewed the data from cycle 2. In cycle 4 the robot reused the network to return to the end point of cycle 2 and continue exploring while the data from cycle 3 were reviewed.*

$6 - 10$, then another cycle idle, then $0.5 - 1$ cycle as in week two, cycle 10. Following this example, and including getting stuck and returning to the lander, we get $63 - 75$ command cycles to do the same work as done in 43 (47% to 77% more cycles). Recall that without NRP, it is unlikely that the robot could have returned to the lander in a single cycle; if we discount the cycles the robot spent at the lander, we get 35 cycles for NRP, and $56 - 64$ cycles for the serial approach (60% to 83% more cycles).

Considering that many of the parallel techniques were being developed and refined during the mock mission, we expect that further operator experience will only increase the improvements made possible by NRP. It should be noted that the length of the command cycle has a strong influence on the efficiency improvement that can be achieved. For example, we predict that even more parallel exploration could have been done if the command cycles were longer (or the robot was faster).

## 6.3 A Robotic Astronaut-Assistant using a Network of Reusable Paths

To this point our discussion has primarily focused on the application of NRP to purely robotic sample return. However, we have also done a second mission scenario using a robot as an astronaut assistant [64]. As this was a mock mission, the astronaut role was not filled by a real astronaut, but instead by a person familiar with both the geology and engineering concerns of the mock mission (for ease of reading we refer to this person as simply the astronaut). This scenario is briefly discussed and some of the beneficial techniques for using the robot are highlighted.

### 6.3.1 Overview of the Astronaut-Assistant Scenario

The robot was at times manually operated by a mock astronaut on site (as seen in Figure 6.10), and at other times it was operated in the same way as described earlier in this chapter. This allowed the system to efficiently leverage the expertise of the astronaut in order to reach sites of interest quickly, and then let the astronaut leave while mission control remotely operated the rover. The rover was equipped with a high-framerate

**Figure 6.10:** *The robot being operated as an astronaut assistant. The mock astronaut was able to take manual control of the robot, or let it be controlled by operators on Earth.*

Autonosys lidar (as in the field trials in Chapter 5) that allowed for lighting-invariant operations. Over the course of the scenario (which consisted of eight days of operations), the robot traveled over 8.2 km. The network that was created was 1.25 km in length, and of that length, 0.92 km were taught by the astronaut manually driving the robot. The remaining 0.33 km was added using the same technique as in the Sudbury mission (i.e., mission control set waypoints relative to the network and the robot used the onboard planning and terrain assessment to attempt to reach them). This means that of the 8.2 km that were driven, approximately 7.3 km were driven autonomously.

### 6.3.2 Techniques for Efficient use of the Robotic Assistant

One of the abilities that first made the NRP robot attractive as an astronaut assistant was that it allowed for some of the best aspects of manned and robotic exploration to be used simultaneously. It allowed us to combine the astronaut's valuable ability to identify interesting areas quickly and respond to unexpected discoveries, with the robot's ability to carry heavier loads and operate for long periods of time.

For example, the astronaut would identify a site of interest, and drive the robot to that site, thereby teaching the robot precisely where to go. The astronaut could do this for many places, and without stopping to take any extended measurements at any of

the sites. The robot could then return, under the direction of mission controllers, and continue with the methodical down-selection process. In the mean time, the astronaut would do other tasks such as explore sites that might be inaccessible to the robot, or the astronaut would eat, sleep, or return to base.

As we carried out the operations, there were further refinements to the techniques that were used to operate the robot. For instance, the astronaut would instruct the rover to return to a previous point on the network (or possibly to a waypoint off the network, though this was not done during any of our tests) and while the rover did the traverse the astronaut would scout the area for other targets to investigate and plan the path for the rover.

We also saw how the astronaut and the robot could complement each other when traveling into new areas. The robot could carry heavy loads (e.g., equipment for sampling, or the samples themselves) and the astronaut could modify terrain to let the rover pass more easily (e.g., the astronaut could fill in a hole or move a rock that made the path more difficult than necessary). This terrain modification was not limited solely to modifying the traversability; the astronaut could also place localization aids. These aids could be rocks with a distinctive appearance in an area that was sparse in visual landmarks. Alternatively, the astronaut could have carried something with which to paint distinctive marks. This is essentially the same as making cairns or placing markers to act as landmarks for a trail that would otherwise be difficult to follow.

## 6.4 Other applications of NRP

NRP can also be used to extend the window for many types of opportunistic investigations. For example, if the robot were to drive past an interesting site, but the site was not identified as interesting until much later, the robot can at a later time, with a single command, return precisely to that previous position. In essence, it provides an insurance policy against leaving a site before all science can be extracted from the data. It also allows for operations during communications blackouts.

NRP could offer benefits for crewed ground vehicles as well. It could allow the vehicle to return astronauts quickly to previous sites, and in the same way as in the astronaut-

assistant scenario, newly driven paths would be added to network as they were driven. This then provides what could be a very valuable safety system for the astronauts; using NRP, the vehicle could automatically return the crew to the base in the event of an emergency. A similar safety system could also benefit purely robotic systems both in planetary exploration and in terrestrial applications. In fact, we can look to the use of robots in the response to the Fukushima-Daiichi accident [132, 133] to see a potential application. In preparation for use in the reactor buildings, the Quince robot was modified to use a physical cable for communication rather than the original wireless control, which would not function in the heavily shielded and high-radiation environment. However, during operation this cable failed and communication with the robot was lost. The robot has not yet been recovered. However, one might imagine a NRP robot having a simple rule that would command the robot to return to base if communication is unexpectedly lost for a long duration. It is possible that had Quince used NRP and had this simple behaviour, it may have been quickly recovered, repaired, and re-deployed. A planetary rover might have similar rules for certain failure conditions, such as: (i) in the event of a prolonged communications outage, automatically return to the last site of successful communication, (ii) in the event of low power reserves and no overriding command (for solar powered robots), automatically return to the last point of high power-generation, or more generally (iii) in the event of certain types of failures of anomalies, automatically return to the last known safe-hold spot and await further commands.

NRP can be used for other applications as well. For example, a work-site mapping scenario [134, 135] has also been carried out[2]. In this, NRP was used onboard the robot as part of the GN&C system. A photo from this test is shown in Figure 6.11.

The image in Figure 6.11 shows the distinctive tracks left by a robot using NRP. Even though the robot has driven these paths many times, the surrounding soil is relatively undisturbed. NRP allows a robot to reduce its environmental impact because it reuses the paths that it has already driven. This may be an important consideration for future missions as it might aid in the protection of the natural sites, and it might help protect the integrity of the data that are collected at those sites.

[2]A video from these tests is available at: http://youtu.be/APGswfxKqEw

**Figure 6.11:** *NRP has been used for work site mapping [134, 135]. This is a photo of a test conducted at the Canadian Space Agency's Mars Emulation Terrain. The network of reusbale paths is visible as tracks in the soft soil.*

## 6.5 Challenges and Future Work

One of the byproducts of using NRP is that the robot tends to travel a greater distance in a shorter time span. This raises two questions: (i) is there sufficient power available to do this additional driving, and (ii) is the additional wear due to more driving compatible with the mission lifetime? Both of these concerns are addressed in rover design. The question of power largely disappears when sources other than solar are used (for example, an onboard reactor). In cases of operation in permanently shadowed regions, such as at the Lunar South Pole, the use of solar power is likely not viable anyway. Even in cases of limited power, it is likely that parallel exploration can be useful by scaling the problem to something that fits within the power budget. Outcrop characterization is one such scenario (see site A in Figure 6.3). In this case the rover might move along the base of an outcrop and collect imagery at many points. It would then return to specific points and continue the down-selection process. In this scenario the rover may only move a few meters between sites of interest, but these different sites would give different vantage points and allow the rover to reach a larger area in less time.

The additional wear could be factored into the design of the rover, or the parallelism could, as above, be scaled to fit within the desired reliability requirements. This scaling is a trade off available to mission controllers.

It should also be noted that NRP offers benefits beyond parallel exploration, such as more accurate goal acquisition and more robust navigation [59, 60], and being able to return to a previous point with one command. These benefits remain even when parallel exploration is not carried out. There are cases in previous missions where having the ability to return automatically to a previously visited point may have saved the operators from the slow process of manually sending incremental movement commands (see Figure 6.1). It seems that similar situations will arise with the Mars Science Laboratory (MSL). At the time of writing, the rover is attempting to reach the Glenelg destination (see Figure 6.12). The name Glenelg is a palindrome, so chosen because the next destination for the rover is almost in the direction directly opposite from the landing site (to the left in the image). This will likely necessitate that MSL return along its outbound path. It is possible that many command cycles and many sols of driving could be eliminated (leaving more time available for scientific tasks) if the rover had the ability to automatically return along this previous paths to return to a previous point. NRP allows just that.

We also need to consider the question of what happens when the vehicle is not able to repeat a path. In all of our testing, this has happened only rarely, and by using a lighting-invariant sensor such as a high-framerate lidar, we can eliminate the inability to localize due to lighting changes or lack of illumination [103, 111]. Additionally, in the case of transient appearance change (e.g., change in the lighting), the robot can simply remain stationary until the conditions return to ones that make the scene appear as when it was taught. This was done as part of mock ground-ice prospecting mission to Mars [136], which used VT&R. However, there will still be cases where the appearance of the scene changes, or the traversability of a previous path changes, and this leads to the desire to be able to repair paths; this is an area of ongoing future work.

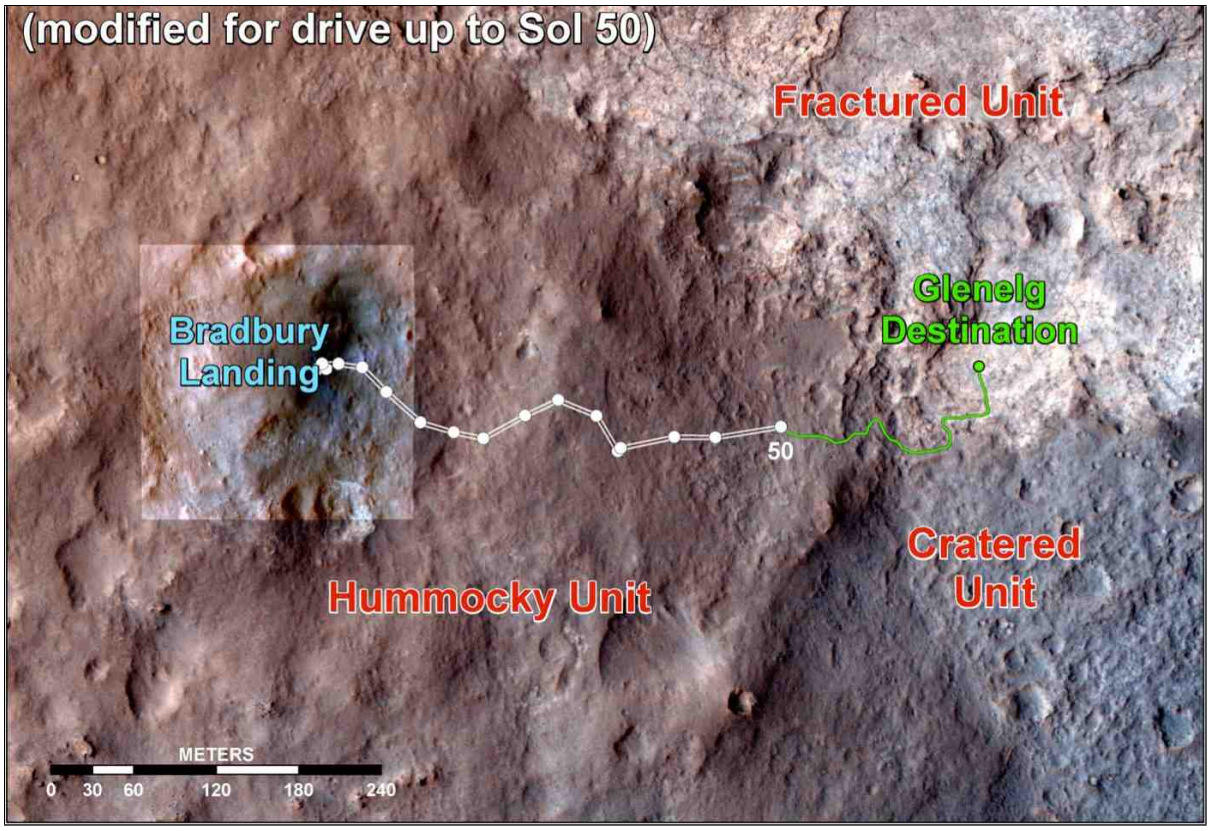


**Figure 6.12:** *The Mars Science Lab approaching Glenelg. The destination after Glenelg is in the direction the almost completely opposite from the landing site. NRP could allow MSL to return quickly and autonomously to a previous point, rather than having the robot receive incremental commands to manually retrace its path. This would leave more time available for scientific tasks. Credit: NASA/JPL-Caltech/University of Arizona*

## 6.6 Conclusions

A network of reusable paths offers a new approach to planetary surface exploration using a mobile robot. NRP has many benefits in the context of robust autonomous navigation [59, 60]. It also allows mission-level improvements by allowing parallel exploration of multiple scientific targets, and it inherently includes sample return and the ability to carry out operations during communication blackouts. During the sample-return analogue mission, this capability enabled nearly twice as many sites to be visited within the mission time frame. Such a capability would be extremely useful for sample-return missions to the Moon or Mars, and likely, many other missions that might make use of a mobile robot.

# Chapter 7

# Final Remarks

The works in this thesis have been presented under the unifying theme of path/action planning for a mobile robot. Path/action planning is an approach that allows a robot to use the onboard planner to decide when to employ a specific GN&C technique from the suite of available GN&C techniques. This final chapter presents a summary of the main contributions in this dissertation. Following that, we identify the main areas that offer promising directions for future works.

## 7.1 Summary of Contributions

The following is a summary of contributions made in this thesis. The Second-Opinion Planner (which was published in the proceedings of a full-paper refereed conference [56] and a journal article [57]) is a path/action-planning framework that makes use of a hierarchy of terrain-assessment techniques. These techniques range from low fidelity at low cost, to high fidelity at high cost. SOP is a method to decide which assessment method to use with specific data from the large amount of raw terrain data that are collected by the robot. In Part I, the SOP algorithm was developed and then tested in simulation and on data from a robot driving more than 9.8 km in a planetary analogue site. The work on SOP offered the following three main contributions:

1. *the identification of the cost of terrain assessment as an important element in the cost of a path,*

2. *a path-planning framework (SOP) that considers the cost of terrain assessment during path planning, and*
3. *results from field trials of a demonstration system using the SOP framework.*

Part II presented a network of reusable paths (which was first published in the proceedings of a full-paper refereed conference [59], and has been submitted for publication as a journal article [60]). In the NRP work, the planner was able to use either of two localization methods: dead-reckoning through visual odometry, or matching against previous local maps through visual teach and repeat. A network of reusable paths is a simple SLAM system, that when combined with the proposed RRT-based path planner, can become a physical embodiment of an RRT. NRP allows for more robust operations and reduced localization error at the goal because pose error is only accumulated along the final path from the goal definition frame to the goal.

As part of the NRP contributions, we presented two sets of field trials carried out at planetary analogue sites. The first used a robot equipped with a stereo camera; the second used a robot equipped with a high-framerate lidar. The vehicle traveled a total of more than 14.4 km in these two tests.

The field trials were carried out during the same deployments as two mock Lunar exploration missions [62, 63]. NRP offers a new paradigm to planetary exploration that allows for parallel science investigations. In the first mock mission, the rover drove a total of 3.9 km on two networks with a total length of approximately 0.7 km. Using NRP allowed for nearly twice as many sites of interest to be visited in the same time as compared to a serial approach to exploration. In the second mock mission, the rover drove a total of 8.2 km on a single network with a length of over 1.2 km. The results from the first mock mission have been accepted to appear as part of conference proceedings [63].

The contributions associated with NRP, as presented in Part II, are as follows:

1. *the development of the concept of a network of reusable paths,*
2. *an RRT-based path-planning algorithm that uses a network of reusable paths in order to seek distant goals in unknown terrain,*
3. *two examples of field robots that used NRP to seek distant goals in natural terrain,*

4. *the introduction, made possible by NRP, of the concept of parallel exploration of sites of interest (as NRP allows the robot to return to any previously visited point),*
5. *the development and field testing of multiple approaches to parallel exploration with a mobile robot, including those that made use of variable autonomy,*
6. *the identification of a methodical down-selection process that is enabled by parallel exploration.*

## 7.2 Future Work

Further development of the Second-Opinion Planner may be at the point of diminishing returns. SOP is already quite near the theoretically best possible performance. There are opportunities for improvements to the SOP implementation, and beyond that, the option for a probabilistic formulation. In fact, many of the design decisions made in the development of SOP were done considering a later probabilistic extension of the framework.

The development of the concept of a network of reusable paths, however, has led to many potential follow-on investigations and improvements. The most promising of these are listed below.

1. *The ability to do path repair.* This would allow the robot to recover in this event that a path becomes untraversable. Path repair also appears to share common aims with loop closure.
2. *The ability to do batch pose estimation to refine the pose estimate of the nodes in the network.* For instance, by adding absolute orientation measurements relative to the network (e.g., from inclinometers, sun sensors, or star trackers), the robot may be able to seek more distant goals because the purely relative localization estimate will not deteriorate as rapidly. This ability may also be desired by operators who wish to see a globally consistent map of the environment, even though such a map is never needed by the robot.
3. *A physical embodiment of an optimal path-planning algorithm.* Some promising candidates are an embodiment of an RRG or RRT*. We predict that given an

optimal planner, the final path to the goal would be the optimal path.

4. *NRP using an underlying teach-and-repeat capability that is adaptive so that it is automatically learning new paths and landmarks as necessary.* The use of distinct operation modes in VT&R simplified the implementation and made the initial tests possible, but the distinct modes ultimately lead to problems when trying, but failing, to repeat previous paths.
5. *Better human-robot interfaces that leverage the possible cooperation between the operator and the robot.* NRP has been demonstrated to offer a very promising interaction between the operator and the robot. This has only seen preliminary development and it is likely that further refinements can be made.
6. *Further development of NRP for mission scenarios.* Other exploration and exploitation missions may benefit from the abilities of NRP.

# Bibliography


[1] A. Makarenko, S. Williams, F. Bourgault, and H. Durrant-Whyte, "An experiment in integrated exploration," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 1, 2002, pp. 534 – 539 vol.1.

[2] P. Furgale and T. Barfoot, "Visual Teach and Repeat for Long-Range Rover Autonomy," *Journal of Field Robotics*, vol. 27(5), p. 534560, September 2010.

[3] S. B. Goldberg, M. W. Maimone, and L. Matthies, "Stereo Vision and Rover Navigation Software for Planetary Exploration," in *In Proceedings of IEEE Aerospace Conference 2002*, Big Sky, Montana, 2002.

[4] J. J. Biesiadecki and M. W. Maimone, "The Mars Exploration Rover surface mobility flight software driving ambition," in *2006 IEEE Aerospace Conference*. Big Sky, MT, USA: Jet Propulsion Lab., California Inst. of Technol., Pasadena, CA, USA, 2006, p. 15.

[5] M. W. Maimone, P. Leger, and J. Biesiadecki, "Overview of the Mars Exploration Rovers' Autonomous Mobility and Vision Capabilities," in *IEEE International Conference on Robotics and Automation*, 2007.

[6] L. Matthies, M. W. Maimone, A. E. Johnson, Y. Cheng, R. G. Willson, C. Villalpando, S. B. Goldberg, A. Huertas, A. N. Stein, and A. Angelova, "Computer Vision on Mars," *International Journal of Computer Vision*, vol. 75, no. 1, pp. 67–92, 2007.

[7] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the Mars Exploration Rovers," *Journal of Field Robotics, Special Issue on Space Robotics*, vol. vol.2, no.3, pp. 169–186, 2007.

[8] J. J. Biesiadecki, P. C. Leger, and M. W. Maimone, "Tradeoffs Between Directed and Autonomous Driving on the Mars Exploration Rovers," *International Journal of Robotics Research*, vol. 26(1), pp. 91–104, 2007.

[9] 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(4), pp. 337–357, 2009.

[10] A. H. Mishkin, J. Morrison, T. Nguyen, H. Stone, B. Cooper, and B. Wilcox, "Experiences with Operations and Autonomy of the Mars Pathfinder Micro-Rover," in *Proceedings of the 1998 IEEE Aerospace Conference*, Snowmass at Aspen, Colorado, March 21-28 1998.

[11] H. W. Stone, "Mars Pathfinder Microrover: A Low-Cost, Low-Power Spacecraft," in *Proceedings of the 1996 AIAA Forum on Advanced Developments in Space Robotics*, Madison, WI, August 1996.

[12] J. Morrison and T. Nguyen, "On-Board Software for the Mars Pathfinder Microrover," in *Proceedings of the Second IAA International Conference on Low-Cost Planetary Missions*, John Hopkins University Applied Physics Laboratory, Laurel, Maryland, April 1996.

[13] S. Kassel, "Lunokhod-1 Soviet Lunar Surface Vehicle," RAND Corporation, Tech. Rep. R-802-ARPA, 1971.

[14] W. Carrier, *Soviet Rover Systems*, ser. LGI TR. Lunar Geotechnical Institute, 1992. [Online]. Available: http://books.google.ca/books?id=JobXHAAACAAJ

[15] R. Volpe, "Rover Functional Autonomy Development for the Mars Mobile Science Laboratory," in *Proceedings of the 2003 IEEE Aerospace Conference*, vol. 2-643, 2003.

[16] B. C. Clark, "Mars sample return: The critical next step," *Acta Astronautica*, vol. 61, no. 16, pp. 95 – 100, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0094576507000264

[17] J. L. Burke, R. R. Murphy, M. D. Coovert, and D. L. Riddle, "Moonlight in Miami: Field Study of Human-Robot Interaction in the Context of an Urban Search and Rescue Disaster Response Training Exercise," *HumanComputer Interaction*, vol. 19, no. 1-2, pp. 85–116, 2004. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/07370024.2004.9667341

[18] K. Nagatani, Y. Okada, N. Tokunaga, S. Kiribayashi, K. Yoshida, K. Ohno, E. Takeuchi, S. Tadokoro, H. Akiyama, I. Noda, T. Yoshida, and E. Koyanagi, "Multirobot exploration for search and rescue missions: A report on map building in RoboCupRescue 2009," *J. Field Robotics*, vol. 28, no. 3, pp. 373–387, 2011.

[19] G. Shaffer and A. Stentz, "A robotic system for underground coal mining," in *IEEE International Conference on Robotics and Automation*, 12-14 May 1992.

[20] J. Marshall, T. Barfoot, and J. Larsson, "Autonomous Underground Tramming for Center-Articulated Vehicles," *Journal of Field Robotics*, vol. 25, no. 6-7, pp. 400–421, June-July 2008.

[21] M. Monta, N. Kondo, and Y. Shibano, "Agricultural robot in grape production system," in *IEEE International Conference on Robotics and Automation*, vol. 3, 1995, pp. 2504 – 2509.

[22] M. M. Foglia and G. Reina, "Agricultural robot for radicchio harvesting," *Journal of Field Robotics*, vol. 23, no. 6-7, pp. 363–377, 2006. [Online]. Available: http://dx.doi.org/10.1002/rob.20131

[23] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The robot that won the DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006. [Online]. Available: http://dx.doi.org/10.1002/rob.20147

[24] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.-W. Seo, S. Singh, J. Snider, A. Stentz, W. . Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson, "Autonomous driving in urban environments: Boss and the Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008. [Online]. Available: http://dx.doi.org/10.1002/rob.20255

[25] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," in *Proceedings of the 19th national conference on Innovative applications of artificial intelligence - Volume 2*, ser. IAAI'07. AAAI Press, 2007, pp. 1752–1759. [Online]. Available: http://dl.acm.org/citation.cfm?id=1620113.1620125

[26] H. R. Everett and D. W. Gage, "From Laboratory to Warehouse: Security Robots Meet the Real World," *The International Journal of Robotics Research*, vol. 18, no. 7, pp. 760–768, 1999. [Online]. Available: http://ijr.sagepub.com/content/18/7/760.abstract

[27] D. Nistr, O. Naroditsky, and J. Bergen, "Visual Odometry for Ground Vehicle Applications," *Journal of Field Robotics*, vol. 23(1), pp. 3–20, 2006.

[28] M. Agrawal and K. Konolige, "Real-time Localization in Outdoor Environments using Stereo Vision and Inexpensive GPS," in *In proceedings of the 18th International Conference on Pattern Recognition (ICPR)*, 2006.

[29] A. Mourikis and S. Roumeliotis, "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation," in *IEEE International Conference on Robotics and Automation*, April 2007.

[30] J.-P. Tardif, M. George, M. Laverne, A. Kelly, and A. Stentz, "A new approach to vision-aided inertial navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.

[31] A. Lambert, P. Furgale, T. D. Barfoot, and J. Enright, "Field testing of visual odometry aided by a sun sensor and inclinometer," *Journal of Field Robotics*, vol. 29, no. 3, pp. 426–444, 2012. [Online]. Available: http://dx.doi.org/10.1002/rob.21412

[32] H. Durrant-Whyte and T. Bailey, "Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms," *IEEE Robotics and Automation Magazine*, vol. 11, no. 3, pp. 99–110, 2006.

[33] T. Bailey and H. Durrant-Whyte, "Simultaneous Localisation and Mapping (SLAM): Part II State of the Art," *IEEE Robotics and Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[34] S. Thrun, "Simultaneous Localization and Mapping," in *Robotics and Cognitive Approaches to Spatial Mapping*, ser. Springer Tracts in Advanced Robotics, M. Jefferies and W.-K. Yeap, Eds. Springer Berlin / Heidelberg, 2008, vol. 38, pp. 13–41.

[35] H. Choset and K. Nagatani, "Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization," *IEEE Transactions on Robotics and Automation*, vol. 17(2), pp. 125–137, 2001.

[36] S. Thrun, "Robotic Mapping: A Survey," in *Exploring Artificial Intelligence in the New Millenium*, G. Lakemeyer and B. Nebel, Eds. Morgan Kaufmann, 2002.

[37] M. Bosse, P. Newman, J. Leonard, and S. Teller, "Simultaneous Localization and Map Building in Large-Scale Cyclic Environments Using the Atlas Framework," *International Journal of Robotics Research*, vol. 23, pp. 1113–1139, 2004.

[38] C. Estrada, J. Niera, and J. D. Tardos, "Hierarchical SLAM: Real-Time Accurate Mapping of Large Environments," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 588–596, 2005.

[39] A. Howard, G. S. Sukhatme, and M. J. Matarić, "Multirobot Simultaneous Localization and Mapping Using Manifold Representations," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1360–1369, 2006.

[40] P. Piniés, L. M. Paz, D. Gálvez-López, and J. D. Tardós, "CI-Graph simultaneous localization and mapping for three-dimensional reconstruction of large and complex environments using a multicamera system," *Journal of Field Robotics*, vol. 27, no. 5, pp. 561–586, 2010.

[41] G. Sibley, C. Mei, I. Reid, and P. Newman, "Vast-scale Outdoor Navigation Using Adaptive Relative Bundle Adjustment," *The International Journal of Robotics Research*, vol. 29-8, pp. 958–980, 2010.

[42] R. Castano, T. Estlin, R. C. Anderson, D. M. Gaines, A. Castano, B. Bornstein, C. Chouinard, and M. Judd, "Oasis: Onboard autonomous science investigation system for opportunistic rover science," *Journal of Field Robotics*, vol. 24, no. 5, pp. 379–397, 2007. [Online]. Available: http://dx.doi.org/10.1002/rob.20192

[43] M. Woods, A. Shaw, D. Barnes, D. Price, D. Long, and D. Pullan, "Autonomous science for an ExoMars Roverlike mission," *Journal of Field Robotics*, vol. 26, no. 4, pp. 358–390, 2009. [Online]. Available: http://dx.doi.org/10.1002/rob.20289

[44] A. Hait, T. Simeon, and M. Taix, "Algorithms for rough terrain trajectory planning," *Advanced Robotics*, vol. 16, no. 7, pp. 673–699, 2002.

[45] C. Ye and J. Borenstein, "T-transformation: traversability analysis for navigation on rugged terrain," *Proceedings of SPIE*, vol. SPIE-5422, pp. 473–483, 2004.

[46] A. R. Green, D. Rye, and H. Durrant-Whyte, "Vehicle Planning in Unstructured Environments," in *Infotech@Aerospoace*, vol. 2005-7097. Arlington, VA: AIAA, 26-29 Sept 2005.

[47] T. M. Howard and A. Kelly, "Optimal Rough Terrain Trajectory Generation for Wheeled Mobile Robots," *International Journal of Robotics Research*, vol. 26, no. 2, pp. 141–166, February 2007.

[48] D. Helmick, A. Angelova, and L. Matthies, "Terrain Adaptive Navigation for Planetary Rovers," *Journal of Field Robotics*, vol. 26(4), pp. 391–410, 2009.

[49] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions of Systems Scinence and Cybernetics*, vol. SSC-4, pp. 100–107, 1968.

[50] D. Ferguson and A. Stentz, "Using Interpolation to Improve Path Planning: The Field D* Algorithm," *Journal of Field Robotics*, vol. 23(2), pp. 79–101, 2006.

[51] S. M. LaValle and J. J. Kuffner, "Randomized Kinodynamic Planning," *The International Journal of Robotics Research*, vol. 20, pp. 378–400, 2001.

[52] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially Constrained Mobile Robot Motion Planning in State Lattices," *Journal of Field Robotics*, vol. 26(3), pp. 308–333, 2009.

[53] S. Karaman and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, June 2011.

[54] F. Ingrand, S. Lacroix, S. Lemai-Chenevier, and F. Py, "Decisional Autonomy for Planetary Rovers," *Journal of Field Robotics*, vol. 24(7), pp. 559–580, 2007.

[55] B. Nabbe and M. Hebert, "Extending the Path-Planning Horizon," *The International Journal of Robotics Research*, vol. 26(10), pp. 997–1024, 2007.

[56] B. Stenning and T. D. Barfoot, "Path Planning with Variable-Fidelity Terrain Assessment," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, 18-22 October 2010, pp. 300–306.

[57] B. E. Stenning and T. D. Barfoot, "Path planning with variable-fidelity terrain assessment," *Robotics and Autonomous Systems*, vol. 60, no. 9, pp. 1135–1148, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889012000826

[58] T. D. Barfoot, P. T. Furgale, B. E. Stenning, P. J. F. Carle, J. P. Enright, and P. Lee, "Devon Island as a Proving Ground for Planetary Rovers," in *Brain, Body, and Machine: Proceedings of an International Symposium on the Occasion of the 25th Anniversary of the McGill University Centre for Intelligent Machines*, ser. Advances in Intelligent and Soft Computing 83, J. Angeles, B. Boulet, J. Clark, J. Kovecses, and K. Siddiqi, Eds. Montreal, Quebec: Springer, 10-12 November 2010, pp. 269–281.

[59] B. Stenning and T. D. Barfoot, "Path Planning on a Network of Paths," in *Proceedings of the IEEE Aerospace Conference*, Big Sky, MT, 5-12 March 2011.

[60] B. Stenning, C. McManus, and T. D. Barfoot, "Planning using a Network of Reusable Paths: A Physical Embodiment of an RRT," 2012, sumbitted to the International Journal of Robotics Research, Special Issue on Motion Planning for Physical Robots, June 15, 2012.

[61] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[62] B. Stenning, G. R. Osinski, T. D. Barfoot, G. Basic, M. Beauchamp, M. Daly, H. Dong, R. Francis, P. Furgale, J. Gammell, N. Ghafoor, A. Lambert, K. Leung, M. Mader, C. Marion, E. McCullough, C. McManus, J. Moores, and L. Preston, "Planetary Surface

Exploration Using a Network of Reusable Paths," in *Proceedings of the 43rd Lunar and Planetary Science Conference (LPSC)*, no. 2360, Texas, USA, 19-23 March 2012.

[63] B. Stenning, G. Osinski, T. Barfoot, G. Basic, M. Beauchamp, M. Daly, R. Francis, P. Furgale, J. Gammell, N. Ghafoor, P. Jasiobedzki, A. Lambert, K. Leung, M. Mader, C. Marion, E. McCullough, C. McManus, J. Moores, and L. Preston, "Planetary Surface Exploration using a Network of Reusable Paths: A Paradigm for Parallel Science Investigations," in *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, 2012.

[64] C. L. Marion, G. R. Osinski, S. Abou-Aly, I. Antonenko, T. Barfoot, N. Barry, A. Bassi, M. Battler, M. Beauchamp, M. Bondy, S. Blain, R. Capitan, E. Cloutis, L. Cupelli, A. Chanou, J. Clayton, M. Daly, H. Dong, L. Ferrire, R. Flemming, L. Flynn, R. Francis, P. Furgale, J. Gammell, A. Garbino, N. Ghafoor, R. A. F. Grieve, K. Hodges, M. Hussein, P. Jasiobedzki, B. L. Jolliff, M. C. Kerrigan, A. Lambert, K. Leung, M. M. Mader, E. McCullough, C. McManus, J. Moores, H. Ng, C. Otto, A. Ozaruk, A. E. Pickersgill, A. Pontefract, L. J. Preston, D. Redman, H. Sapers, B. Shankar, C. Shaver, A. Singleton, K. Souders, B. Stenning, P. Stooke, P. Sylvester, J. Tripp, L. L. Tornabene, T. Unrau, D. Veillette, K. Young, and M. Zanetti, "A Series of Robotic and Human Analogue Missions in Support of Lunar Sample Return," in *LPS XLIII*, 2012.

[65] R. Francis, G. R. Osinski, J. Moores, T. Barfoot, and I. Team, "Co-operative Human-Robotic Exploration of Lunar Analogue Sites," in *Proceedings of the 43rd Lunar and Planetary Science Conference (LPSC)*, no. 1996, Texas, USA, 19-23 March 2012.

[66] J. E. Moores, R. Francis, M. Mader, G. Osinski, T. Barfoot, N. Barry, G. Basic, M. Battler, M. Beauchamp, S. Blain, M. Bondy, R.-D. Capitan, A. Chanou, J. Clayton, E. Cloutis, M. Daly, C. Dickinson, H. Dong, R. Flemming, P. Furgale, J. Gammel, N. Gharfoor, M. Hussein, R. Grieve, H. Henrys, P. Jaziobedski, A. Lambert, K. Leung, C. Marion, E. McCullough, C. McManus, C. Neish, H. Ng, A. Ozaruk, A. Pickersgill, L. Preston, D. Redman, H. Sapers, B. Shankar, A. Singleton, K. Souders, B. Stenning, P. Stooke, P. Sylvester, and L. Tornabene, "A Mission Control Architecture for robotic lunar sample return as field tested in an analogue deployment to the sudbury impact structure," *Advances in Space Research*, no. 0, pp. –, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0273117712003183

[67] S. Singh, R. Simmons, T. Smith, A. Stentz, V. Verma, A. Yahja, and K. Schwehr, "Recent Progress in Local and Global Traversability for Planetary Rovers," in *Proceedings of the 2000 IEEE International Conference on Robotics & Automation*, San Francisco, CA, April 2000.

[68] C. Leger, A. Trebi-Ollennu, J. Wright, S. Maxwell, R. Bonitz, J. Biesiadecki, F. Hartman, B. Cooper, E. Baumgartner, and M. Maimone, "Mars Exploration Rover Surface Operations: Driving Spirit at Gusev Crater," in *In Proceedings of the 2005 IEEE Conference on Systems, Man, and Cybernetics*, 2005.

[69] J. Biesiadecki, E. Baumgartner, R. Bonitz, B. Cooper, F. Hartman, P. Leger, M. Maimone, S. Maxwell, A. Trebi-Ollennu, E. Tunstel, and J. Wright, "Mars Exploration Rover Surface Operations: Driving Opportunity at Meridiani Planum," in *IEEE International Conference on Systems, Man, and Cybernetics*, 2005.

[70] A. Stentz, "Optimal and Efficient Path Planning for Unknown and Dynamic Environments," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-93-20, 1993.

[71] S. Koenig, "Fast Replanning for Navigation in Unknown Terrain," *IEEE Transactions on Robotics*, vol. 21(3), pp. 354–363, 2005.

[72] A. Stentz, "The Focussed D* Algorithm for Real-Time Replanning," in *In Proceedings of the International Joint Conference on Artificial Intelligence*, 1995, pp. 1652–1659.

[73] A. Yahja, S. Singh, and A. Stentz, "An efficient on-line path planner for outdoor mobile robots," *Robotics and Autonomous Systems*, vol. 32, pp. 129–143, 2000.

[74] D. Cagigas, "Hierarchical D* algorithm with materialization of costs for robot path planning," *Robotics and Autonomous Systems*, vol. 52, pp. 190–208, 2005.

[75] D. Ferguson and A. T. Stentz, "The Delayed D* Algorithm for Efficient Path Replanning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, April 2005, pp. 2045 – 2050.

[76] G. A. Mills-Tettey, A. T. Stentz, and M. B. Dias, "DD* Lite: Efficient Incremental Search with State Dominance," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-07-12, May 2007.

[77] J. Carsten, A. Rankin, D. Ferguson, and A. Stentz, "Global Path Planning on Board the Mars Exploration Rovers," in *In IEEE Aerospace Conference 2007*, Big Sky, MT, March 2007, pp. 1–11.

[78] J. Gancet and S. Lacroix, "PG2P: a perception-guided path planning approach for long range autonomous navigation in unknown natural environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003).*, vol. 3, 2003, pp. 2992– 2997.

[79] D. Ferguson and A. Stentz, "Planning with imperfect information," in *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, vol. 2, Sept.-2 Oct. 2004, pp. 1926–1931 vol.2.

[80] R. W. Floyd, "Algorithm 97: Shortest Path," Communications of the ACM 5 (6): 345, June 1962.

[81] S. Warshall, "A theorem on Boolean matrices," *Journal of the ACM*, vol. 9 (1), p. 1112, January 1962.

[82] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Eds., *Introduction to Algorithms (2nd ed.)*. Cambridge, Massachusetts: MIT Press, 2001, ch. All-Pairs Shortest Paths, p. 620642.

[83] A. Fournier, D. Fussell, and L. Carpenter, "Computer rendering of stochastic models," *Communications of the ACM*, vol. 25, no. 6, pp. 371–384, 1982.

[84] S. Kambhampathi and L. S. Davis, "Multiresolution Path Planning for Mobile Robots," *IEEE Journal of Robotics and Automation*, vol. (RA-2)3, pp. 135–145, 1986.

[85] S. Behnke, "Local Multiresolution Path Planning," in *in Proc. of 7th RoboCup Int. Symposium*, Padua, Italy, 2003.

[86] R. V. Cowlagi and P. Tsiotras, "Multiresolution Path Planning with Wavelets: A Local Replanning Approach," in *American Control Conference*, Westin Seattle Hotel, Seattle, Washington, USA, June 11-13 2008.

[87] P. J. Carle, P. T. Furgale, and T. D. Barfoot, "Long-range rover localization by matching LIDAR scans to orbital elevation maps," *Journal of Field Robotics*, vol. 1-27, pp. 1–27, 2010.

[88] M. Likhachev, D. Ferguson, G. Gordon, A. T. Stentz, and S. Thrun, "Anytime Dynamic A*: An Anytime, Replanning Algorithm," in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, June 2005.

[89] K. Belghith, F. Kabanza, L. Hartman, and R. Nkambou, "Anytime Dynamic Path-Planning with Flexible Probabilistic Roadmaps," in *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, Orlando, Florida, May 2006.

[90] C. Tovey and S. Koenig, "Localization: Approximation and Performance Bounds to Minimize Travel Distance," *IEEE Transactions on Robotics*, vol. 26-2, pp. 320–330, 2010.

[91] R. Li, K. Di, L. H. Matthies, R. E. Arvidson, W. M. Folkner, and B. A. Archinal, "Rover Localization and Landing-Site Mapping Technology for the 2003 Mars Exploration Rover Mission," *Photogrammetric Engineering & Remote Sensing*, vol. 70(1), pp. 77–90, 2004.

[92] C. Olson, L. Matthies, M. Schoppers, and M. Maimone, "Rover Navigation Using Stereo Ego-motion," *Robotics and Autonomous Systems*, vol. vol. 43(4), pp. pp. 215–229, 2003.

[93] R. Volpe, "Navigation Results from Desert Field Tests of the Rocky 7 Mars Rover Prototype," *The International Journal of Robotics Research*, vol. 18-7, pp. 669–683, 1999.

[94] R. Chatila and J.-P. Laumond, "Position Referencing and Consistent World Modeling for Mobile Robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, Mar. 1985, pp. 138–145.

[95] R. C. Smith, M. Self, and P. Cheeseman, "Estimating Uncertain Spatial Relationships in Robotics," in *Autonomous Robot Vehicles*, I. J. Cox and G. T. Wilfong, Eds. New York: Springer Verlag, 1990, pp. 167–193.

[96] P. Hébert, S. Betgé-Brezetz, and R. Chatila, "Decoupling odometry and exteroceptive perception in building a global world map of a mobile robot: the use of local maps," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, Apr. 1996, pp. 757–764.

[97] K. S. Chong and L. Kleeman, "Feature-Based Mapping in Real, Large Scale Environments Using an Ultrasonic Array," *International Journal of Robotics Research*, vol. 18, no. 1, pp. 3–19, 1999.

[98] S. Simhon and G. Dudek, "A Global Topological Map formed by Local Metric Maps," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Victoria, Canada, 1998.

[99] B. Kuipers, J. Modayil, P. Beeson, M. MacMahon, and F. Savelli, "Local Metrical and Global Topological Maps in the Hybrid Spatial Semantic Hierarchy," in *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, New Orleans, LA, 2004.

[100] A. Howard, "Multi-robot mapping using manifold representations," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, vol. 4, New Orleans, LA, 2004, pp. 4198–4203.

[101] K. Konolige, J. Bowman, J. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua, "View-based Maps," *The International Journal of Robotics Research*, vol. 29-8, pp. 941–957, 2010.

[102] R. Brooks, "Visual Map Making for a Mobile Robot," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, Mar. 1985, pp. 824–829.

[103] C. McManus, P. Furgale, B. Stenning, and T. D. Barfoot, "Visual Teach and Repeat using appearance-based lidar," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, may 2012, pp. 389 –396.

[104] A. M. Zhang and L. Kleeman, "Robust Appearance Based Visual Route Following for Navigation in Large-scale Outdoor Environments," *The International Journal of Robotics Research*, vol. 28-3, pp. 331–356, 2009.

[105] Y. Matsumoto, M. Inaba, and H. Inoue, "Visual navigation using view-sequenced route representation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, 1996, pp. 83–88.

[106] T. Ohno, A. Ohya, and S. Yuta, "Autonomous navigation for mobile robots referring pre-recorded image sequence," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, 1996, pp. 672–679.

[107] S. Jones, C. Andresen, and J. Crowley, "Appearance based processes for visual navigation," in *Proceedings of the IEEE Int. Conference on Intelligent Robots and Systems*, 1997.

[108] Y. Matsumoto, K. Sakai, M. Inaba, and H. Inoue, "View-based approach to robot navigation," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, Takamatsu, Japan, 2000, pp. 1702–1708.

[109] L. Tang and S. Yuta, "Vision based navigation for mobile robots in indoor environment by teaching and playing-back scheme," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3, Seoul, Korea, 2001, pp. 3072–3077.

[110] E. Royer, M. Lhuillier, M. Dhome, and J. Lavest, "Monocular vision for mobile robot localization and autonomous navigation," *International Journal of Computer Vision*, vol. 74(3), 2007.

[111] T. Barfoot, B. Stenning, P. Furgale, and C. McManus, "Exploiting Reusable Paths in Mobile Robotics: Benefits and Challenges for Long-term Autonomy," in *9th Canadian Conference on Computer and Robot Vision (CRV)*, Toronto, Canada, 28-30 May 2012, pp. 388–395.

[112] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, June 1981. [Online]. Available: http://doi.acm.org/10.1145/358669.358692

[113] A. Hait, T. Simeon, and M. Taix, "Robust motion planning for rough terrain navigation," in *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Piscataway, NJ; USA, 1999, pp. 11–16 vol.1.

[114] N. Roy, W. Burgard, D. Fox, and S. Thrun, "Coastal navigation-mobile robot navigation with uncertainty in dynamic environments," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 1, 1999, pp. 35 –40 vol.1.

[115] J. Gonzalez and A. Stentz, "Planning with uncertainty in position an optimal and efficient planner," in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, 2-6 2005, pp. 2435 – 2442.

[116] N. Melchior and R. Simmons, "Particle RRT for Path Planning with Uncertainty," in *Robotics and Automation, 2007 IEEE International Conference on*, 10-14 2007, pp. 1617 –1624.

[117] R. Pepy, M. Kieffer, and E. Walter, "Reliable robust path planner," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, 22-26 2008, pp. 1655 –1660.

[118] B. Bonet and H. Geffner, "Planning with Incomplete Information as Heuristic Search in Belief Space," in *Proceedings of the 6th International Conference on Artificial Intelligence in Planning Systems (AIPS)*. AAAI Press, 2000, pp. 52–61.

[119] S. Prentice and N. Roy, "The Belief Roadmap: Efficient Planning in Belief Space by Factoring the Covariance," *The International Journal of Robotics Research*, vol. (28) 11-12, pp. 1448–1465, 2009.

[120] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959, 10.1007/BF01386390. [Online]. Available: http://dx.doi.org/10.1007/BF01386390

[121] T. Howard, C. Green, A. Kelly, and D. Ferguson, "State Space Sampling of Feasible Motions for High Performance Mobile Robot Navigation in Complex Environments," *Journal of Field Robotics*, vol. Vol 25, No 6-7, pp. pp. 325–345, 2008.

[122] C. Urmson and R. Simmons, "Approaches for heuristically biasing RRT growth," in *Proceedings of the IEEE/RSJ International Conference on Robotics and Systems (IROS)*, 2003.

[123] D. Ferguson and A. Stentz, "Anytime RRTs," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*, 2006.

[124] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *Open-Source Software workshop of the International Conference on Robotics and Automation (ICRA)*, 2009.

[125] W. Churchill and P. Newman, "Practice Makes Perfect? Managing and Leveraging Visual Experiences for Lifelong Navigation," in *Proc. IEEE International Conference on Robotics and Automation (ICRA2012)*, Minnesota, USA, May 2012.

[126] A. J. Lambert, "Visual Odometry Aided by a Sun Sensor and an Inclinometer," Master's thesis, University of Toronto, 2011.

[127] P. T. Furgale, T. D. Barfoot, and G. Sibley, "Continuous-Time Batch Estimation Using Temporal Basis Functions," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, St. Paul, USA, 14-18 May 2012, pp. 2088–2095.

[128] C. H. Tong, P. T. Furgale, and T. D. Barfoot, "Gaussian Process Gauss-Newton: Non-Parametric State Estimation," in *9th Canadian Conference on Computer and Robot Vision (CRV)*, Toronto, Canada, 28-30 May 2012, pp. 206–213.

[129] H. J. Dong and T. D. Barfoot, "Lighting-Invariant Visual Odometry using Lidar Intensity Imagery and Pose Interpolation," in *Proceedings of the International Conference on Field and Service Robotics (FSR), to appear.*, Matsushima, Japan, 16-19 July 2012.

[130] O. Takahashi and R. Schilling, "Motion planning in a plane using generalized voronoi diagrams," *IEEE Transactions on Robotics and Automation*, vol. 5(2), pp. 143–150, 1989.

[131] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12(4), pp. 566–580, 1996.

[132] S. Kawatsuma, M. Fukushima, and T. Okada, "Emergency response by robots to fukushima-daiichi accident: summary and lessons learned," *Industrial Robot: An International Journal*, vol. 39(5), pp. 428 – 435, 2012.

[133] K. Nagatani, S. Kiribayashi, Y. Okada, K. Otake, K. Yoshida, S. Tadokoro, T. Nishimura, T. Yoshida, E. Koyanagi, M. Fukushima, and S. Kawatsuma, "Emergency response to the nuclear accident at the fukushima daiichi nuclear power plants using mobile rescue robots," *Journal of Field Robotics*, pp. n/a–n/a, 2012. [Online]. Available: http://dx.doi.org/10.1002/rob.21439

[134] C. H. Tong, T. D. Barfoot, and E. Dupuis, "Three-dimensional SLAM for mapping planetary work site environments," *Journal of Field Robotics*, vol. 29, no. 3, pp. 381–412, 2012. [Online]. Available: http://dx.doi.org/10.1002/rob.21403

[135] R. S. Merali, C. H. Tong, J. Gammell, J. Bakambu, E. Dupuis, and T. D. Barfoot, "3D Surface Mapping Using a Semi-Autonomous Rover: A Planetary Analog Field Experiment," in *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, Turin, Italy, September 2012.

[136] T. Barfoot, P. Furgale, B. Stenning, P. Carle, L. Thomson, G. Osinski, M. Daly, and N. Ghafoor, "Field Testing of a Rover Guidance, Navigation, & Control Architecture to Support a Ground-Ice Prospecting Mission to Mars," *Robotics and Autonomous Systems*, vol. 59, no. 6, pp. 472–488, 2011.