Sampling-Based Hierarchical Motion Planning for a Reconfigurable Wheel-on-Leg Planetary Analogue Exploration Rover

William Reid *
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA, USA
william.reid@jpl.caltech.edu

Robert Fitch *
School of Mechanical and Mechatronic Engineering
University of Technology Sydney
Sydney, NSW, Australia
rfitch@uts.edu.au

Ali Haydar Göktoğan
Australian Centre for Field Robotics
The University of Sydney
Syndey, NSW, Australia
ali.goktogan@acfr.usyd.edu.au

Salah Sukkarieh
Australian Centre for Field Robotics
The University of Sydney
Syndey, NSW, Australia
salah.sukkarieh@acfr.usyd.edu.au

Abstract

Reconfigurable mobile planetary rovers are versatile platforms that may safely traverse cluttered environments by morphing their physical geometry. Planning paths for these adaptive robots is challenging due to their many degrees of freedom, and the need to consider potentially continuous platform reconfiguration along the length of the path. We propose a novel hierarchical structure for asymptotically optimal (AO) sampling-based planners and specifically apply it to the state-of-the-art Fast Marching Tree (FMT*) AO planner. Our algorithm assumes a decomposition of the full configuration space into multiple sub-spaces, and begins by rapidly finding a set of paths through one such sub-space. This set of solutions is used to generate a biased sampling distribution, which is then explored to find a solution in the full configuration space. This technique provides a novel way to incorporate prior knowledge of sub-spaces to efficiently bias search within existing AO sampling-based planners. Importantly, probabilistic completeness and asymptotic optimality are preserved. Experimental results in simulation are provided that benchmark the algorithm against state-of-the-art sampling-based planners without the hierarchical variation. Additional experimental results performed with a physical wheel-on-leg platform demonstrate application to planetary rover mobility and showcase how constraints such as actuator failures and sensor pointing may be easily incorporated into the planning problem. In minimizing an energy objective that combines an approximation of the mechanical work required for platform locomotion with that required for reconfiguration, the planner produces intuitive behaviours where the robot dynamically adjusts its footprint, varies its height, and clambers over obstacles using legged locomotion. These results illustrate the generality of the planner in exploiting the platform's mechanical ability to fluidly transition between various physical geometric configurations, and wheeled/legged locomotion modes, without the need for predefined configurations.

*Work was performed while authors were affiliated with the Australian Centre for Field Robotics, The University of Sydney.

1 Introduction

Over the past five decades the use of mobile robotic rovers to perform *in-situ* scientific investigations on the surface of Mars has been tremendously influential in shaping our understanding of this extraterrestrial environment (Grotzinger et al., 2014; Arvidson et al., 2010; Arvidson et al., 2006). As robotic missions have evolved, the desire to explore more unstructured terrain has increased. This thinking has exposed mobility limitations with conventional rover designs such as getting stuck in soft soil or simply not being able to access rugged terrain, as demonstrated in efforts to cross mega-ripple sand dunes by the NASA *Curiosity* rover (Arvidson et al., 2017). Additionally, future rovers will need to assist human explorers with surface operations such as haulage and construction during future long-duration human missions to the Moon or Mars (Wilcox et al., 2007). Increased mobility and terrain traversability are key requirements when considering designs for next generation planetary rovers. Coupled with these requirements is the need to autonomously navigate unstructured terrain by taking full advantage of increased mobility.

Reconfigurable wheeled mobile robots (RWMRs) are a class of vehicle that may utilize many degrees of freedom to traverse challenging terrain. This is done by combining wheeled and legged locomotion modes to fluidly transition over, under or around unstructured obstacles by driving pseudo-omnidirectionally in combination with morphing geometric structure. The effectiveness of such a mobility system is demonstrated in the extensive field testing of the SherpaTT rover on a Martian analogue site with significant obstacles and inclines (Cordes et al., 2018). To leverage RWMRs' capabilities within the planetary exploration context, techniques for autonomous decision making within platforms' high-dimensional configuration space are necessary. The major contributions of this article are the development of a motion planner for high-degree-of-freedom (high-DOF) RWMRs in addition to the demonstration of such a planner in both simulated and physical RWMR traversals over challenging terrain.

The main challenge in planning for RWMRs is to efficiently find a feasible low cost path given potentially many degrees of freedom. The problem is to generate a sequence of transitions between initial and terminal system states (configurations) while both avoiding environmental obstacles and satisfying internal kinematic constraints. The sequence of transitions is to be optimized with respect to a cost function that is used to define the objectives of the robot's overall motion. In this article an approximation of the energy expenditure of the robot is minimized.

Asymptotically optimal (AO) sampling-based planners make it feasible to plan paths for high-DOF robots and converge towards an optimal solution. This article describes a modification to existing AO sampling-based planners that both preserves AO and exploits prior knowledge of task structure to promote faster convergence. Further, the proposed planning architecture assists in the search for narrow passageways, a notable problem for many sampling-based planners. This approach enables the use of prior knowledge of kinematic structure to inform the choice of hierarchical decomposition, but defined in a way that is more flexible than in other hierarchical approaches. The state-space decomposition must still be provided *a priori*, but the sub-tasks are continuous and generated probabilistically by the planner. The extent of the configuration space explored by sub-tasks is defined by continuous parameters that may be automatically tuned. The proposed planner is denoted as Hierarchical Bidirectional Fast Marching Trees (HBFMT*).

A main technical challenge in developing a hierarchical sampling-based planner is to define the information flow between levels in a way that does not sacrifice completeness and optimality guarantees. The planner decomposes the configuration space as a hierarchy of increasingly larger sub-spaces, each searched by a Fast Marching Tree (FMT*) (Janson et al., 2015) variant. Solutions from adjacent levels are used to bias search, thereby increasing path quality with equivalent computational resources. This technique is suited to platforms that have intuitively identifiable low-dimensional kinematic structure, such as RWMRs, but may also generalize to other high-DOF robots such as redundant manipulator arms.

It is shown that this approach retains the completeness and asymptotic optimality guarantees of FMT* (Janson et al., 2015) by virtue of a fraction of its samples, ℓ being taken from a uniform distribution. Experimental

(a) Cube-like compact state. (b) Fully deployed single leg raised state. (c) Bridge configuration.

Figure 1: Various configurations of the reconfigurable MAMMOTH rover.

results in simulation show that HBFMT* outperforms FMT*, Bidirectional Fast Marching Trees (BFMT*) (Starek et al., 2016), AO Rapidly-exploring Random Trees (RRT*) (Karaman and Frazzoli, 2011), Informed RRT* (Gammell and Barfoot, 2014) and Batch Informed Trees (BIT*) (Gammell et al., 2015) in a variety of environments.

Use of the planner with a physical RWMR system is also presented in a series of traverses over obstacle-littered terrain. These trials utilize the MAMMOTH (Mars Analogue Multi-MOde Traverse Hybrid) rover as the RWMR. This vehicle is capable of driving omnidirectionally, raising and lowering its ground clearance, and reconfiguring its footprint. Some of the unique configurations of this robot are shown in Fig. 1. For each traverse a safe path is generated for the MAMMOTH rover to follow using the HBFMT* algorithm. In addition to physically validating the HBFMT* planner, capabilities for imposing motion constraints within the planner are demonstrated. In some of the traverses, simulated actuator breakages or platform heading constraints are introduced. By doing so, the versatility in planning for a variety of different configuration spaces is illustrated. These experiments are some of the first globally planned traverses demonstrated with a physical RWMR.

This article expands on the HBFMT* planner first introduced in (Reid et al., 2016a). All simulated and physical experiments presented in this article utilize the MAMMOTH rover system summarized in (Reid et al., 2014) and (Reid et al., 2016b). The article discusses related work in Sec. 2, formulates the motion planning problem and presents background on the FMT* algorithm in Sec. 3. HBFMT* is presented in Sec. 4, while analysis of the algorithm is provided in Sec. 5. Details specific to the MAMMOTH rover planning problem are presented in Sec. 6 in preparation for the numerical experiments presented in Sec. 7 and physical experiments presented in Sec. 8. Conclusions and discussion of future work is provided in Sec. 9.

2 Related Work

In this section, we summarize the state-of-the-art in sampling-based planning as well as the use of hierarchical planning frameworks to constrain large configuration spaces, thereby boosting computational efficiency. The drawbacks of such frameworks, including over-biasing towards sub-spaces and lack of completeness and optimality guarantees, are also discussed. Planning methodologies for real world high-DOF systems are investigated to motivate the need for hierarchical sampling-based planning frameworks to better leverage reconfigurability in the practical setting.

2.1 Sampling-Based Planning

There are two major classes of sampling-based planning algorithms: multi-query and single query. Multi-query planners construct a roadmap of a space that can be queried with different initial and goal states multiple times. Single-query planners solve a planning problem for a single initial and goal state pair. Multi-query sampling-based techniques include the probabilistically complete probabilistic roadmap planner (PRM) (Kavraki et al., 1996) and its AO variant PRM* (Karaman and Frazzoli, 2011). These techniques first sample a batch of random nodes from a space, which are then connected to their nearest neighbours to form a roadmap. This roadmap may then be searched using standard graph-search algorithms such as Dijkstra or A*. PRM* ensures asymptotic optimality by considering edges to nodes within a nearest neighbourhood. The nearest neighbourhood is defined by a distance r_n that is a function of the number of nodes within the current graph, the dimensionality of the planning problem and the volume of the space being explored:

$$r_n \geq 2\left(\frac{1}{d}\right)^{\frac{1}{d}}\left(\frac{\mu(\mathcal{X}_{free})}{\zeta_d}\right)^{\frac{1}{d}}\left(\frac{\log(n)}{n}\right)^{\frac{1}{d}}, \tag{1}$$

where ζ_d is the Lebesgue measure of the unit-cost ball of dimension d and $\mu(\mathcal{X}_{free})$ is the Lebesgue measure of the free space. Eq. 1 is derived in (Janson et al., 2015). Much effort has been focused on developing variants of PRMs that address the computational bottleneck of edge construction (Dobson and Bekris, 2014; Esposito and Wright, 2016; Salzman et al., 2013). Additional improvements include heuristic techniques that lazily check obstacle collisions along promising candidate paths as described for PRM in (Bohlin and Kavraki, 2000) and for PRM* in (Hauser, 2015). Recent work by (Solovey and Kleinbort, 2018) has shown that the connection radius r_n may be reduced even further so that is on the order of $n^{(-1/d)}$ as opposed to $(\log(n)/n)^{(-1/d)}$ and retain an asymptotically (near-) optimal guarantee.

The AO guarantee derived in (Janson et al., 2015) is performed for a Euclidean space. AO guarantees for a general space with a metric cost function are also discussed with minor modifications to Eq. (1) relating to the Lebesgue measure of the space. Further study of AO guarantees for non-Euclidean spaces, specifically kinodynamic spaces, are included in (Schmerling et al., 2015b), (Schmerling et al., 2015a), (Li et al., 2016), (Hauser and Zhou, 2016), (Webb and Van Den Berg, 2013) and (Karaman and Frazzoli, 2010).

Examples of single-query planning algorithms include the RRT (LaValle and Kuffner, 2001) and its asymptotic optimal counterpart RRT* (Karaman and Frazzoli, 2011), the Expansive Space Trees (Hsu, 2000) and the single query bidirectional probabilistic roadmap (SBL) (Sanchez and Latombe, 2003) planners. In the RRT variants a state is sampled from a uniform distribution and a nearest neighbour state within the tree is grown towards the sampled state. On each iteration of the RRT expansion a tree edge growth is attempted and is successful if the edge has no collision with the environment or breaks an internal constraint of the robot model. The algorithm iterates until the tree reaches a goal region or an expiry time is reached.

Single-query AO algorithms include RRT* (Karaman and Frazzoli, 2011), FMT* (Janson et al., 2015) and BIT* (Gammell et al., 2015). Each of these techniques relies on sampling a number of states from the free configuration space and growing a tree through these samples. As with PRM*, a nearest neighbourhood of nodes is evaluated when expanding the tree from a single sample. This nearest neighbourhood is defined by a distance r_n as defined in Eq. (1). Both RRT* and BIT* are anytime planners, attempting to improve a solution once it has been found. FMT* is not an anytime planner, instead relying on a predetermined number of samples and building a tree through these samples. Variants of FMT* that make it anytime are the anytime FMT* and Motion Planning with Lower Bounds algorithms, both presented in (Salzman and Halperin, 2015).

As with the PRM variants, much emphasis has been put on improving the computational efficiency of single query sampling-based planners. In (Arslan and Tsiotras, 2013), the AO RRT$^{\#}$ planner is proposed where tree leaves are characterized based on their potential to be apart of a better solution once an initial solution has been found. Only the leaves that can possibly improve the solution may be re-wired. A similar idea is used in Anytime RRTs (Ferguson and Stentz, 2006), C-FOREST RRT (Otte and Correll, 2013) and Informed

RRT* (Gammell et al., 2014) where once the RRT finds a solution only samples within an informed hyper-ellipse are taken. The ellipse represents a bound on the largest possible cost-to-come plus cost-to-go distance of a node. A promising avenue for increased planning performance is the combination of sampling-based planning and deterministic sampling as opposed to random sampling as presented in (Janson et al., 2018). A common practice within the planning literature for improving computational efficiency is using a k-nearest neighbor (KNN) search as opposed to a radial nearest neighbor (RNN) search. As mentioned in (Janson et al., 2015), KNN search is more adaptable to different obstacle spaces. During RNN search, for a node next to an obstacle, a significant proportion of the neighborhood ball may be inside the obstacle, resulting in a limited number of neighboring samples. In KNN search the neighborhood considered will be the k closest nodes, therefore not limiting the number of samples.

2.2 Hierarchical Sampling-Based Planning

Hierarchical planning is a popular technique used to boost computational efficiency. This is demonstrated by the SyCLoP algorithm detailed in (Plaku et al., 2010). The planner initially decomposes the workspace of the robot into a discrete grid or set of triangular regions, which is then searched using a standard graph search technique. The initial search returns a set of 'lead' regions, which are used to guide a sampling-based planner through the full configuration space of the robot. As the exploration progresses, the corresponding discrete regions are updated based on how successful the exploration is. A similar methodology is used within the Exploration Exploitation Trees planner proposed in (Rickert et al., 2014). At the high-level, the robot workspace is explored using a wavefront expansion 'bubble' planner that finds collision free 'tunnels' through the workspace, which is based on a workspace decomposition planner proposed in (Brock and Kavraki, 2001). The purpose of the high-level planner is to identify a coarse obstacle free structure of the environment. At the low-level, an RRT-Connect algorithm (LaValle and Kuffner, 2001) is used to rapidly perform a bidirectional search of the full configuration space. The RRT-Connect is guided by choosing frontier 'bubbles' that are promising candidate regions for extending the plan towards the goal. As the RRT-Connect progresses, the exploitation score of the associated workspace bubble is updated based on how successful it is in progressing through the bubble.

The authors of both (Plaku et al., 2010) and (Rickert et al., 2014) describe their methodologies as trading off between exploration and exploitation. The high-level workspace plans provide knowledge of the environment that the low-level planners may exploit. If the high-level planner does not progress through certain regions they both have mechanisms by which exploration of new workspace regions is promoted. Neither of these algorithms maintain the probabilistic completeness guarantees of the sampling-based planners they use at the low-level of their hierarchies. Both sets of authors argue that in practice the reliance on exploitation of the workspace plan results in two to three orders of magnitude improvement in computational efficiency as compared with RRT based search of the full configuration space. The HBFMT* algorithm presented in this article exploits knowledge from a lower level sub-space plan and at the same time takes advantage of the FMT* AO guarantee with biased sampling to retain AO and therefore completeness. A shortcoming of hierarchical techniques is seen when the workspace plan does not capture the necessary complexities of moving through a high-dimensional space. An example of such a case is discussed in Section 5. Fortunately, for the reconfigurable wheeled mobile robot planning problem, most navigation cases have the workspace of the rover's body frame embedded within the full configuration space of the robot.

For the hierarchical planners already discussed, a sub-space is selected by intuition. A recent topic of interest is in learning sub-spaces to assist in biasing full configuration space exploration. In (Rowekamper et al., 2013), the planning problem for a planar mobile dual-arm manipulation robot is presented. A low-dimensional configuration space projection is learned from a succession of paths being run between randomly selected start and goal states in an example environment. A sampling-based planner is then run within this projected space. In (Vernaza and Lee, 2012) low-dimensional structure is learned by finding the directions in which the planning cost function varies principally. A set of the most influential principal components together describe basis motions of the robot, which are used to describe the low-dimensional sub-space to plan in. This technique is applied to a manipulation arm as well as a PR2, a dual manipulator arm and

mobile platform system. Most recently, in (Ichter et al., 2018) a sampling distribution is learned using a conditional variational autoencoder, which is then explored by a FMT* planner. Unlike these planners, the HBFMT* planning algorithm assumes a given state variable decomposition, but is the first principled hierarchical sampling-based planner that retains performance guarantees.

The HBFMT* planner leverages the configuration space size reduction enabled by hierarchical planning, however unlike the hierarchical planners presented so far, HBFMT* operates completely in continuous space by using sampling-based planning at each hierarchy level. Ultimately, the performance of discrete grid-based planners is limited by grid resolution and the ability to generate a cost map over the entire configuration space. An added advantage of using a sampling-based planner at the high-level is that the configuration space may be easily modified by introducing a modified cost function, collision detector and sampler. This is more straightforward than using a discrete planner where a new cost-map would have to be generated with each configuration space modification, which may become computationally intractable with an increase in configuration space dimension. Like grid-based planners sampling-based planners are still exponential in computational efficiency, however are much less susceptible to the *curse of dimensionality.*

2.3 Real World Planning for Reconfigurability

A common planning strategy used for real world high-dimensional legged, tracked or wheel-on-leg robotic systems is to use a sampling-based planner within a planning hierarchy. Generally, these planners take the form of a coarse footfall planner at the high level, and then a sampling-based planner for low-level leg swing and body posture planning.

A recent hierarchical planner that uses RRT* at each level is presented in (Wermelinger et al., 2016). In (Wermelinger et al., 2016), the three-tiered hierarchical planner, which will be denoted as Hierarchical RRT* (HRRT*), is used to explore the space with successively finer resolution footprint models of the ETHStar quadruped walking robot. The technique lends itself to planning within the coarse global map available to the robot before its traverse with the coarse high-level RRT* planner. Local terrain information gleaned from laser sensors is used to iteratively plan with the lower-level RRT* planners using an increasingly higher resolution model of the platform's footprint. HRRT* is targeted at the anytime planning problem where local sensor information is updated throughout the robot's traverse, whereas HBFMT* is focused on reconfigurable systems and exploiting the various modes of traversing a terrain in a computationally efficient manner.

The highly reconfigurable NASA/JPL Robosimian quadruped is described in (Karumanchi et al., 2016). The platform's four limbs each have seven DOF, resulting in a hyper-manoeuverable walking platform. A hierarchical planner described in (Satzinger et al., 2015) is used to tame the high-dimensionality of the platform by constraining the search space. Initially, a footfall planner returns a global path over unstructured terrain, a body pose search then returns a central body motion that complies with inverse kinematic solutions constrained by leg stances, and finally an RRT-Connect search returns a path for a swing-leg between footfall states constrained by the central body motion. Hierarchical search used to constrain configuration space size is also employed in the hexapod planner described in (Belter et al., 2015) and the articulated tracked vehicle planner in (Brunner et al., 2015).

The NASA/JPL All-Terrain Hex-Limbed Extra-Terrestrial Explorer (ATHLETE) rover, with its 42 actuated joints, also presents a challenging planning and control problem as discussed in (SunSpiral et al., 2012). A hierarchical planning system that combines footfall planning and the SBL sampling-based planner (Sanchez and Latombe, 2003) is used to plan safe leg raising and lowering paths over obstacle-littered terrain. Field testing demonstrated the use of the planner to get around local rock obstacles and also dismount from a mock habitat. Full body motion planning for ATHLETE is proposed in (Hauser et al., 2008), however has only been demonstrated in simulation.

Like the work presenting ATHLETE we present a planner tailored to a high-DOF robot that may be used

to explore planetary analogue environments. This article is amongst the first to demonstrate the use of hierarchical motion planning with asymptotic optimality guarantees with a real world robotic system.

3 Problem Formulation and Background

This section formalizes the planning problem for a high-DOF robotic system. Additionally, the FMT* algorithm (Janson et al., 2015), on which HBFMT* is based, is summarized for convenience.

3.1 Problem Definition

Let $\mathcal{X} = [0,1]^d$ be a configuration space with dimension $d \in \mathbb{N}$ constrained to $d \geq 2$. Let $\mathcal{X}_{obs}$ be the obstacle region such that $\mathcal{X} \setminus \mathcal{X}_{obs}$ is an open set. This implies that the free configuration space is a closed set $\mathcal{X}_{free} = cl(\mathcal{X} \setminus \mathcal{X}_{obs})$. We randomly sample n points from $\mathcal{X}_{free}$. Let $\mathcal{X}^{\#} = [0,1]^f$ be a sub-space that is embedded within $\mathcal{X}$ where $1 \leq f \leq d$. Any variable with the "#" super-script is used to denote a variable related to the sub-space $\mathcal{X}^{\#}$.

A path planning problem is denoted by a triplet $(\mathcal{X}_{free}, x_{init}, \mathcal{X}_{goal})$ where x_{init} is the initial state and $\mathcal{X}_{goal}$ is the goal region. In this work, the goal region is considered as a single state, described by x_{goal}. A path is defined by a continuous mapping $\pi : [0,1] \to \mathbb{R}^d$ such that $0 \mapsto x_1$ and $1 \mapsto x_2$. Let Σ be the set of all paths in $\mathcal{X}$. A feasible path in the planning problem $(\mathcal{X}_{free}, x_{init}, \mathcal{X}_{goal})$ is a path that is collision free with $\pi(0) = x_{init}$ and $\pi(1) = x_{goal}$. $\Pi = \{\pi_0, \pi_1, ..., \pi_k\}$ is a set of k feasible paths sorted according to path cost.

The optimal path planning problem is to find a path π^*, assuming problem $(\mathcal{X}_{free}, x_{init}, \mathcal{X}_{goal})$ and an arc cost function $c : \Sigma \to \mathbb{R}_{\geq 0}$ such that $c(\pi^*) = \min\{c(\pi) : \pi \text{ is feasible}\}$, or to report failure. The optimal path π^* is δ-*robustly feasible* if every point along it is a minimum of δ away from $\mathcal{X}_{obs}$.

3.2 FMT*

The FMT* algorithm, listed in Alg. 1, performs a recursive dynamic programming procedure to simultaneously explore and construct a single-query tree through a configuration space. The algorithm begins by randomly sampling n nodes from a uniform distribution using the `SampleUniform` function and putting them into the set S along with the the initial and goal nodes (lines 1-3). The tree T is then initialized with S and root node $\mathbf{x}_{init}$.

FMT* initialization is listed in Alg. 2. Tree $T = (V, E, H, W)$ is composed of tree nodes V, and edges E. The open set of nodes is denoted as H, and W is the unvisited set. The tree nodes V are initialized as all of the nodes in S, there are initially no edges in E, the nodes in the unvisited set to begin with are everything but the root node x_o, while the only node in the open set is x_o .

The tree is then grown outward using a wavefront expansion, always finding the optimal solution to the current node with respect to the available nodes and the prescribed metric cost function. After FMT* initialization in Alg. 1 the tree expansion procedure is run until the goal node has been found, returning the path π, or no goal has been found with no path returned (lines 5-11).

FMT* expansion, listed in Alg. 3, is performed by maintaining two sets of nodes: an open and an unvisited set. More concisely, tree $T = (V, E, H, W)$ is composed of tree nodes V, and edges E. The open set of nodes is denoted as H, and W is the unvisited set. Tree T is initialized with n nodes sampled from free space, $\mathcal{X}_{free}$, in addition to the start node, $\underline{x}_{start}$ and goal node, $\underline{x}_{goal}$. The algorithm maintains an open set, H, and an unvisited set, W. Initially, $\underline{x}_{start}$ is placed in H, while every other node as well as the goal node is placed in W. The node set, V is initialized containing $\underline{x}_{start}$, while the edge set E is initialized as empty.

Algorithm 1: Fast Marching Tree (FMT*) (Janson et al., 2015)

Data: $\mathcal{X}_{free}$,$\underline{x}_{init}$,$\underline{x}_{goal}$,n
Result: π,T
1 $success = false$
2 $stop\_expansion = false$
3 $S \leftarrow \underline{x}_{init} \cup \underline{x}_{goal} \cup$ `SampleUniform`(n)
4 $T \leftarrow$ `Initialize`$(S,\underline{x}_{init})$
5 $\underline{z} \leftarrow \underline{x}_{init}$
6 **while** $stop\_expansion = false$ and $success = false$ **do**
7 $\quad \{T, \underline{z}, sucess, stop\_expansion\} \leftarrow$ `Expand`$(T,\emptyset,\underline{z},\emptyset)$
8 **if** $success = true$ **then**
9 $\quad \pi \leftarrow$ `Path`$(\underline{x}_{init},\underline{x}_{goal},T)$
10 **else**
11 $\quad \pi \leftarrow \emptyset$

Algorithm 2: FMT* Initialize

1 **function** `Initialize`$(S,\underline{x}_o)$
2 $\quad V \leftarrow S$, $E \leftarrow \emptyset$, $W \leftarrow V \setminus \{\underline{x}_o\}$, $H \leftarrow \{\underline{x}_o\}$
3 $\quad$ **return** $T = (V, E, H, W)$

The algorithm then performs a recursive dynamic programming procedure that propagates a wavefront of state transitions (tree edges) outwards from the start node.

The expansion procedure starts by taking the lowest cost node $\underline{z}$ within the open set H and finding its nearest neighbourhood Z_{near} of nodes that are also within the unvisited set W (line 2). The Z_{near} neighbourhood is iterated over and connections are attempted between each of these nodes and the closest nodes to them that are also in the open set X_{near}. A connection to a node $\underline{x}_{min}$ in the nearest neighbourhood is attempted only once; if a collision is detected on this candidate connection, no more attempts will be made to connect $\underline{x}_{min}$ to any other node in the open set during this expansion iteration (lines 3-12). This is termed *lazy collision detection*. Once all nearest neighbourhood nodes have been iterated over, all of the newly connected nodes are added to the tree, taken out of the unvisited set and placed in H. The original lowest cost node within the open set is also removed from the open set, never to be re-visited (line 13). Lastly, a new minimum open set node $\underline{z}$ is assigned in preparation for the next expansion iteration (line 16). This expansion procedure continues until the goal node is successfully found, or the open set is empty, implying that no nodes are left to be expanded into and failure is returned. Like PRM* and RRT*, the nearest neighbourhood in FMT* is defined by the radial distance given in Eq. (1).

FMT* is advantageous in that it uses lazy collision detection to reduce the number of expensive collision checking operations, and converges to optimal solutions faster than RRT* and PRM* (shown experimentally in (Janson et al., 2015) and (Starek et al., 2014)). Like RRT* and PRM*, FMT* is proven to be AO (Janson et al., 2015). The proof may be modified to allow for non-uniform sampling distributions. The HBFMT* planner presented in this article also exploits non-uniform sampling by biasing solutions found from FMT* searches in sub-dimensional spaces. It is shown that the conditions for AO are preserved in Section 5.

3.3 BFMT*

In general, the usage of bidirectional search within the planning problem may dramatically reduce the amount of required exploration (LaValle, 2006). Additionally, bidirectional search has been shown to aid in the discovery of narrow passageways for sampling-based planners (Starek et al., 2014; Jordan and Perez,

Algorithm 3: FMT* Expansion

function `Expand`$(T = (V, E, H, W), T' = (V', E', H', W'), \underline{z}, \underline{x}_{collision})$

1 $H_{new} \leftarrow \emptyset$
2 $Z_{near} \leftarrow$ `Near` $(V \setminus \{\underline{z}\}, \underline{z}, r_n) \cap W$
3 **for** $\underline{x} \in Z_{near}$ **do**
$X_{near} \leftarrow$ `Near`$(V \setminus \{\underline{x}\}, \underline{x}, r_n) \cap H$
$\underline{x}_{min} \leftarrow \text{argmin}_{\underline{x}' \in X_{near}} \{$`Cost`$(\underline{x}, T)$ + `Cost`$(\overline{\underline{x}'\underline{x}})\}$
4 **if** `CollisionFree`$(\underline{x}_{min}, \underline{x})$ **then**
5 $E \leftarrow E \cup \{(\underline{x}_{min}, \underline{x})\}$
6 $H_{new} \leftarrow H_{new} \cup \{\underline{x}\}$
7 $W \leftarrow W \setminus \{\underline{x}\}$
8 $T = (V, E)$
9 **if** $\underline{x} \in V'$ and `Cost`$(\underline{x}, T)$ + `Cost`$(\underline{x}, T')$ < `Cost`$(\underline{x}_{collision}, T)$ + `Cost`$(\underline{x}_{collision}, T')$ **then**
10 $\underline{x}_{collision} \leftarrow \underline{x}$
11 **if** $\underline{x}_{min} = \underline{x}_{goal}$ **then**
12 $success = true$
13 $H \leftarrow (H \cup H_{new}) \setminus \{\underline{z}\}$
14 **if** $H = \emptyset$ **then**
15 $stop\_expansion = true$
16 $\underline{z} \leftarrow \text{argmin}_{\underline{x} \in H} \{$`Cost`$(\underline{x}, T)\}$
17 **return** $\{T = (V, E, H, W), \underline{z}, success, stop\_expansion, \underline{x}_{collision}\}$

2013). Within the hierarchical planning framework proposed in this article, bidirectional search is employed for this reduction in exploration while searching $\mathcal{X}_{free}$. It is also used to find multiple candidate paths while searching $\mathcal{X}^{\#}_{free}$ as described in Sec. 4. BFMT* is a bidirectional variant of FMT* proposed in (Starek et al., 2014).

BFMT* grows two FMT*s, $T = (V, E, H, W)$ and $T' = (V', E', H', W')$, rooted at the start and goal nodes respectively. Pseudo-code for this function is presented in Alg. 4 and in (Starek et al., 2014). BFMT* starts by sampling a set of states, S, from $\mathcal{X}_{free}$. The two trees are then initialized. The samples S, the start state $\underline{x}_{init}$ and the goal state $\underline{x}_{goal}$ are first inserted into the V set of T, with $\underline{x}_{init}$ as the root node. The states S, $\underline{x}_{init}$ and $\underline{x}_{goal}$ are also inserted into V' of T', with $\underline{x}_{goal}$ at its root. During each iteration of BFMT* the tree T is expanded (line 7), termination conditions are checked (lines 8-16), and then success criteria are checked (lines 17-20). At the end of an iteration a new minimum cost node z is selected from the alternate tree T' (line 21). The two trees are then swapped (line 22), so that the alternated tree may be expanded on the next iteration of algorithm. Tree expansion ceases once a termination condition has been encountered.

The termination condition of BFMT* may occur in one of two separate conditions (Starek et al., 2014). The first is when the two trees initially intersect, occurring when a node is in the open set of both trees. This is the "first path" termination condition, however it does not guarantee an optimal solution. The second termination condition occurs when a node is in the open set of one tree, while not inside the open or unvisited set of the other. This second "best path" condition results in a case where the node can no longer improve its cost-to-come value from either tree root and is therefore the optimal solution given the sampled nodes. When one of these termination conditions are encountered the path from π, which is the path from the initial goal of the first tree T to the collision node $\underline{x}_{collision}$, and then to the initial goal T' (lines 8-10 of Alg. 4).

Algorithm 4: Bidirectional Fast Marching Trees

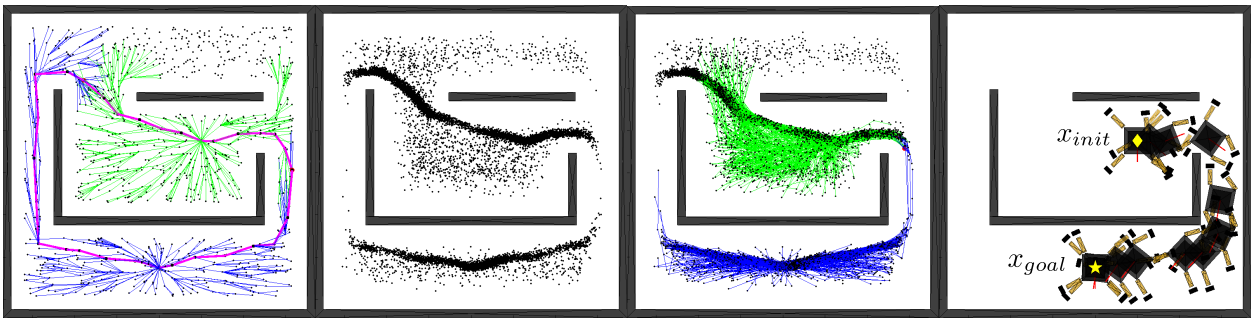
function BFMT*($\mathcal{X}_{free}$,$\underline{x}_{init}$,$\underline{x}_{goal}$,n,Π,ℓ,r_{singe},r_{tunnel},tc)

$S \leftarrow \underline{x}_{init} \cup \underline{x}_{goal} \cup$ SampleFree $(n, \Pi, \ell, r_{tunnel})$
$T \leftarrow$ Initialize$(S, \underline{x}_{init})$
$T' \leftarrow$ Initialize$(S, \underline{x}_{goal})$
$\Pi \leftarrow \emptyset$, $\underline{z} \leftarrow \underline{x}_{init}$, $\underline{x}_{collision} \leftarrow \emptyset$
$e \leftarrow false$, $success \leftarrow false$,$stop\_expansion \leftarrow false$
while $stop\_expansion = false$ and $success = false$ **do**
  $\{T, \underline{z}, success, stop_expansion\, x_{collision}\} \leftarrow$Expand$(T,T',\underline{z},success,\underline{x}_{collision})$
  if ($tc =$ FIRST and $\underline{x}_{collision} \neq \emptyset$) or ($tc =$ BEST and $\underline{z} \in (V' \setminus H')$) **then**
    $\pi \leftarrow$ Path$(\underline{x}_{collision},T) \cup$ Path$(\underline{x}_{collision},T')$
    $\Pi \leftarrow \Pi \cup \pi$, $success = true$
  else if $tc =$ EXHAUSTIVE and $\underline{z} \in (V' \setminus H')$ **then**
    $\pi \leftarrow$ Path$(x_{collision},T) \cup$ Path$(\underline{x}_{collision},T')$
    $\Pi \leftarrow \Pi \cup \pi$
    SingeBranch$(\underline{x}_{collision},T,r_{singe})$
    SingeBranch$(\underline{x}_{collision},T',r_{singe})$
    $\underline{x}_{collision} \leftarrow \emptyset$, $e \leftarrow true$
  if $H = \emptyset$ and $H' = \emptyset$ and $e = false$ **then**
    $stop\_expansion = true$
  else if $H = \emptyset$ and $H' = \emptyset$ and $e = true$ **then**
    $success = true$
  $\underline{z} \leftarrow \text{argmin}_{\underline{x}' \in H'}\{$Cost$(\underline{x}',T')\}$
  Swap(T,T')
return Π

4 Hierarchical Bidirectional Fast Marching Trees (HBFMT*)

Intuitively, HBFMT* biases search of the full configuration space based on knowledge gathered from an initial rapid search of a sub-space. To introduce HBFMT* we describe an example with the MAMMOTH rover constrained to an 8-DOF space $\mathcal{X}$ operating within a confined environment shown in Fig. 2. This space is composed of the 3-DOF position of the robot with respect to an inertial frame (x_B^I, y_B^I, z_B^I), the yaw of the robot ψ and leg $i's$ hip rotation for each of the four legs q_{H_i}. The robot starts in an open-stance state at the diamond position, while its goal is to achieve a similar stance at the star position both shown in Fig. 2d. Any feasible path requires reconfiguration of the robot to either traverse over and around block obstacles or clamber over step obstacles. Initially, as in Fig. 2a, a sub-space $\mathcal{X}_{free}^{\#}$ is searched using BFMT*. The sub-space describes the translation of the robot's body with respect an inertial frame. Collision checking is performed between the environment and the translating body of the robot without its legs. A set of paths, $\Pi^{\#}$, is composed of feasible paths through $\mathcal{X}_{free}^{\#}$ and is shown in Fig. 2a. The full 8-DOF configuration space of the robot is then sampled with a bias around the set of returned paths through $\mathcal{X}_{free}^{\#}$ as shown in Fig. 2b. Another BFMT* instance then searches this biased distribution as in Fig. 2c. The bias is implemented using a tunable *tunnel radius*, r_{tunnel} that defines how focused the search of the solutions from the previous level of the hierarchy is. A final path that traverses around and over the box obstacles is returned and shown in Fig. 2d.

Before HBFMT* is defined, a modification to the termination condition of BFMT* is described. An "exhaustive" termination condition is introduced; pseudo-code is provided in Alg. 4. When a "best" or "first" termination condition is found at a tree collision node $x_{collision,i}$, BFMT* returns the feasible path π_i, where i is the path index, and adds it to the set of alternative paths $\Pi^{\#}$. When a collision node is found it is singed.



(a) The BFMT* in $\mathcal{X}^{\#}_{free}$. The two returned paths make up the set $\Pi^{\#}$.

(b) The samples in $\mathcal{X}_{free}$ projected into $\mathbb{R}^2$.

(c) The BFMT* in $\mathcal{X}_{free}$ projected into $\mathbb{R}^2$.

(d) The MAMMOTH rover at various waypoints along the path.

Figure 2: In the HBFMT* algorithm an initial exhaustive BFMT* search (a) of a uniformly sampled sub-space returns a set of feasible paths. States from the full space are then sampled from a biased distribution focused around the sub-space paths (b). A second BFMT* shown in (c) searches through the full state samples. Lastly, in (d) a full state-space path is returned.

Algorithm 5: Hierarchical Bidirectional Fast Marching Trees

Data: $\mathcal{X}_{free}$,$\underline{x}_{init}$,$\underline{x}_{goal}$,n,r_{singe},r_{tunnel}
Result: Π

1 **begin**
2 $\quad\Pi^{\#} \leftarrow$ BFMT*$(\mathcal{X}^{\#}_{free},\underline{x}^{\#}_{init},\underline{x}^{\#}_{goal},n,\emptyset,1,r_{singe},r_{tunnel},EXHAUSTIVE)$
3 $\quad\Pi \leftarrow$ BFMT*$(\mathcal{X}_{free},\underline{x}_{init},\underline{x}_{goal},n,\Pi^{\#},\ell,\emptyset,r_{tunnel},BEST)$

All of its descendant nodes are found via a breadth first search starting from the collision node. All nodes within a distance r_{singe} of the collision node or its descendants are removed from the open and unvisited sets so that tree expansion may not continue through them. This singeing procedure is performed in both FMT*s. Pseudo-code for the singeing procedure is presented in Alg. 7. BFMT* continues this search and singe procedure until both trees' open sets are exhausted.

HBFMT* is now formalized. Pseudo-code is listed in Alg. 5. The first step is to plan a path through the sub-space $\mathcal{X}^{\#}_{free}$ using an exhaustive BFMT*, sampling uniformly from $\mathcal{X}^{\#}_{free}$. Depending on the space $\mathcal{X}^{\#}_{free}$ and the performance of the BFMT* search, there may be multiple unique paths returned to form the set $\Pi^{\#}$.

The set of paths $\Pi^{\#}$ is then passed to a second BFMT* instance along with a sampling variable ℓ. This second BFMT* search starts by sampling n states from $\mathcal{X}_{free}$ to create the tree nodes V as detailed in Alg. 6. A collection of $n\ell$ samples are taken from a uniform distribution over $\mathcal{X}_{free}$ to comply with asymptotic optimality conditions discussed in Sec. 5. Every second path within $\Pi^{\#}$ is reversed and then all of the paths within $\Pi^{\#}$ are concatenated to form a path λ that goes back and forth between the goal and end nodes (lines 2-7). The sub-space components of $n(1-\ell)$ states are sampled from a Gaussian distribution $\mathcal{N}(d_\lambda, r_{tunnel})$ where d_λ is a uniformly sampled distance along λ (line 11). The parameter r_{tunnel}, the tunnel radius, determines how wide the biased search region around λ is. Currently, this variable is tuned manually, however in future work it is desired that this variable based on results from multiple searches of $\mathcal{X}^{\#}_{free}$. The remaining sub-space components are sampled uniformly (line 13). A sample is only added to the set S if it is valid, having no collisions with obstacles, respects kinematic constraints and is within the configuration space boundaries (lines 14-15). An example of a resulting set of sampled nodes in $\mathcal{X}_{free}$ projected into $\mathbb{R}^2$ is shown in Fig. 2b. Once the biased set of samples S has been found, the second BFMT* instance explores the set and returns the best path through the resulting tree as shown in Fig. 2d.

Algorithm 6: Sample Free

```
function SampleFree(n,Π,ℓ,r_tunnel)
    λ ← [ ]
    for i = 1 to |Π| do
        if i mod 2 = 0 then
            λ ← [λ  Reverse(Π_i)]
        else
            λ ← [λ  Π_i]
    m ← n
    while m > 0 do
        if m < (1 − ℓ)n then
            x ← SampleBiased(λ,r_tunnel)
        else
            x ← SampleUniform
        if IsValid(x) then
            m ← m − 1, S ← S ∪ x
    return S
```

Algorithm 7: Singe Branch

```
function SingeBranch(z,T,r_singe)
    /* Perform breadth-first search of the sub-tree with its root at node z.  */
    A ← BreadthFirstSearch(z,T)
    A ← A∪ Near(T,A,r_singe)
    H ← H \ A, V ← V \ A
    H′ ← H′ \ A, V′ ← V′ \ A
```

5 Analysis

In this section the asymptotic optimality of HBFMT* is shown and the effects of its tuning parameters are discussed. Asymptotic optimality (and thus, probabilistic completeness) of HBFMT* is proved by observing that the terminal stage of the algorithm is BFMT* with a non-uniform sampling distribution and a metric cost function. The following theorem characterizes asymptotic optimality in terms of the number of sample nodes.

Theorem 5.1. *Let $\pi : [0,1]$ be a feasible path with strong δ-clearance, $\delta > 0$. ζ is the Lebesgue measure of the unit-cost ball and $\mu(\mathcal{X}_{free})$ is the Lebesgue measure of the free space. Consider running HBFMT* by sampling n nodes from a distribution φ and executing to completion using any termination criteria and a radius*

$$r_n = 2(1+\eta)\left(\frac{1}{d}\right)^{\frac{1}{d}}\left(\frac{\mu(\mathcal{X}_{free})}{\zeta}\right)^{\frac{1}{d}}\left(\frac{\log(n\ell)}{n}\right)^{\frac{1}{d}}\left(\frac{1}{\ell}\right)^{\frac{1}{d}}, \tag{2}$$

for a parameter $\eta \geq 0$ and $n\ell > 1$. Let c_n denote the cost of the path returned by HBFMT and c^* be the optimal path cost, then $\lim_{n\to\infty} \mathbb{P}(c_n > (1+\varepsilon)c^*) = 0$ for all $\varepsilon > 0$ (which defines AO).*

Proof. The terminal stage of HBFMT* implements BFMT* operating in $\mathcal{X}_{free}$, using a metric cost function and samples from probability distribution φ that is lower bounded by ℓ. To prove AO of HBFMT* it therefore suffices to show that BFMT* is AO under these two conditions. Firstly, any metric cost function satisfies the AO conditions for FMT* (Sec. 5 of (Janson et al., 2015)). Secondly, we can represent φ as a mixture distribution, composed of a uniform distribution that is sampled with probability ℓ and a non-uniform distribution that is sampled with probability $1-\ell$ (Janson et al., 2015). This results in a uniform

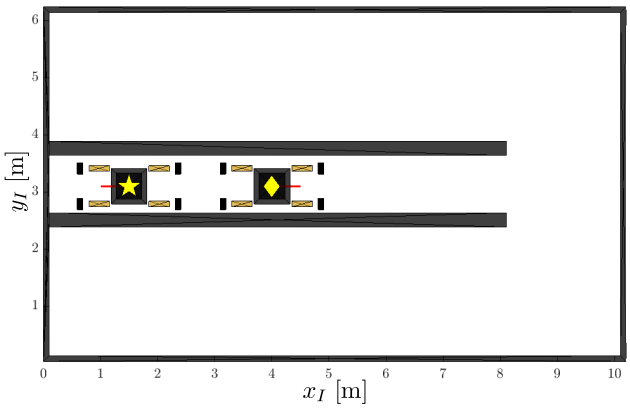


Figure 3: The "Spin to Win" environment in which the MAMMOTH rover must drive out of the narrow passageway, change its yaw angle by 180°, and then drive back into the narrow passageway. The start state is at the diamond location, while the goal state is at the star location.

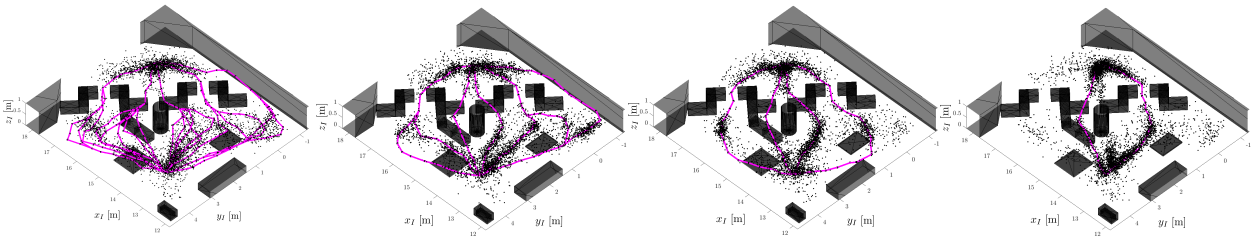
distribution being sampled $n\ell$ times, and thus the proof of AO of FMT* is preserved. Likewise, for BFMT*, $\lim_{n\to\infty} \mathbb{P}(c_n > (1+\varepsilon)c^*) = 0$ for all $\varepsilon > 0$ (Starek et al., 2014).

□

The performance of HBFMT* compared with BFMT* and FMT* is largely dependent on the sub-space $\mathcal{X}^{\#}_{free}$ that is initially sampled and the resulting distribution φ generated in $\mathcal{X}_{free}$. It is generally the case for reconfigurable mobile robots that $\mathcal{X}^{\#}$ is the workspace of the vehicle, which is in the translational $\mathbb{R}^3$ Euclidean space. However, if the platform requires a specific orientation or joint articulation motion that is not contained within the biased search region in $\mathcal{X}_{free}$, the algorithm may not perform as well as FMT* and BFMT* and may be dependent on samples taken from the uniform component of φ. Currently, the sub-space $\mathcal{X}^{\#}_{free}$ is chosen *a priori*, however a topic of future work is incorporation of autonomous selection of these spaces given a certain environment and robot model. Examples of such low-dimensional structure identification can be found in (Rowekamper et al., 2013) and (Vernaza and Lee, 2012).

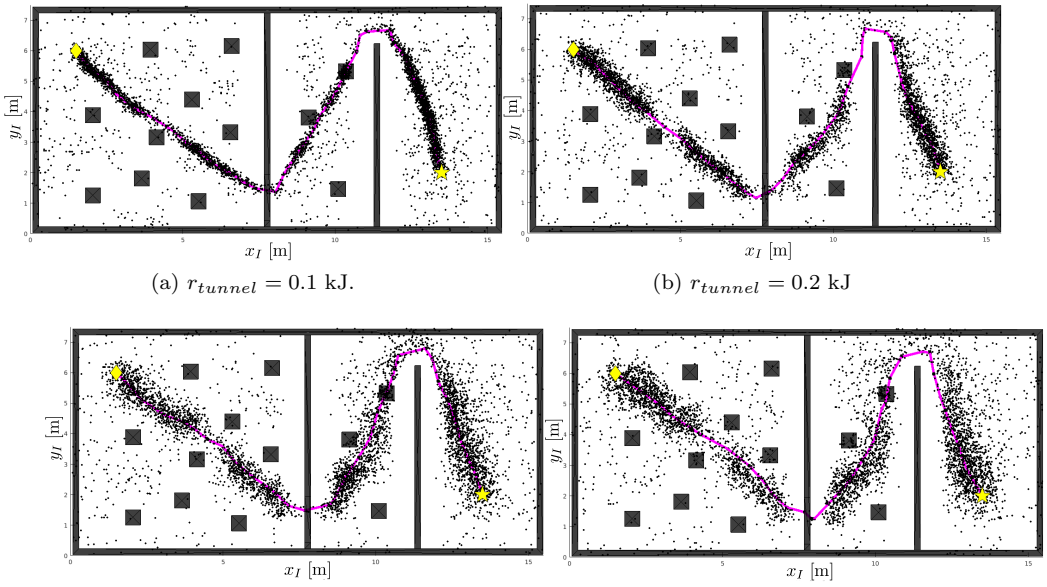
An example environment in which a dominant dimensions outside of $\mathbb{R}^3$ are present is called "Spin to Win" and is shown in Fig. 3. The MAMMOTH rover starts in a narrow passageway, and its goal position is inside the same narrow passageway, however the yaw angle is rotated by 180°. To get to the goal state from the initial state the rover must drive out of the passageway, change its heading and then drive back into the passageway. The use of HBFMT* exploring $\mathbb{R}^3$ in its first hierarchy level would not perform very well given that the yaw dimension is a dominant one within the full planning problem. This problem may be addressed by including the yaw of the robot within $\mathcal{X}^{\#}$. By doing this, a second problem arises. If the body without legs is used for collision detection within the first hierarchy level it will be able to yaw by 180° within the narrow passageway, unlike the full rover model. It therefore becomes apparent that the footprint dimensions are also dominant within this problem and a BFMT* or FMT* that uniformly samples $\mathcal{X}$ may be more useful.

The singe radius, r_{singe} parameter affects the number of alternative paths within $\Pi^{\#}$. By increasing the size of the singe radius a larger sub-set of the two trees' nodes are prohibited from being expanded through. The effect of varying r_{singe} is shown in Fig. 4. The figure illustrates the resulting alternative paths through $\mathcal{X}_{free}$ in an environment with many possible homotopic paths to choose from. As a rule of thumb the r_{singe} parameter should define a ball that is larger than the radius that defines the ball that can completely envelop the body of the robot being planned for. Any smaller value will most likely result in redundant sub-space paths that will implicitly create a sampling bias towards the region of the full space that they lie in. As r_{singe} gets larger the number of alternate paths found will decrease. Therefore, if it is desired that the number



(a) $r_{singe} = 0.5$ m (b) $r_{singe} = 1$ m (c) $r_{singe} = 1.5$ m (d) $r_{singe} = 2$ m

Figure 4: The paths through $\mathcal{X}_{free}^{\#}$ generated by the exhaustive BFMT* with varying r_{singe} values. The samples taken from the biased sampling distribution generated around the $\mathcal{X}_{free}^{\#}$ paths are also shown.



(a) $r_{tunnel} = 0.1$ kJ. (b) $r_{tunnel} = 0.2$ kJ

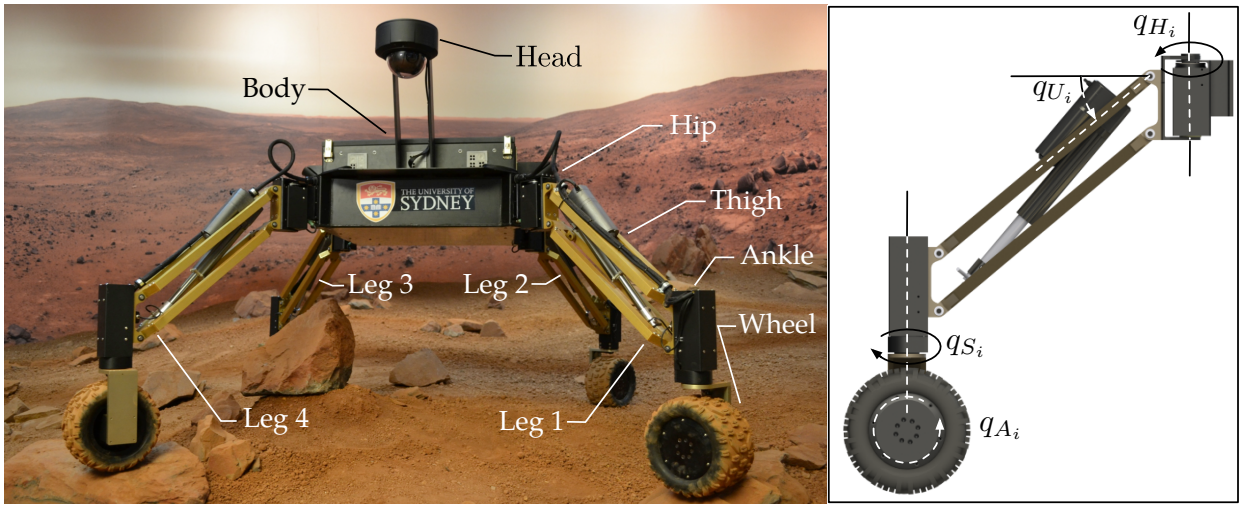
(c) $r_{tunnel} = 0.3$ kJ (d) $r_{tunnel} = 0.4$ kJ

Figure 5: The effect of different tunnel radii sizes on samples in $\mathcal{X}_{free}$ taken from a biased distribution focused around a path found from an exploration of $\mathcal{X}_{free}^{\#}$.

of sub-space paths be smaller so as to focus on a smaller exploration region in the final hierarchy level, the r_{singe} value should be increased. The choice of r_{singe} is a trade-off between a focused sampling distribution in $\mathcal{X}_{free}$ and knowledge of existing sub-space paths embedded within $\mathcal{X}_{free}$.

An additional parameter that affects the performance of HBFMT* is the biased sampling distribution variance r_{tunnel}. A small value for r_{tunnel} results in focused sampling around the nodes in $\Pi^{\#}$, thereby increasing the planner's reliance on the sub-state solutions $\Pi^{\#}$, and making the planner more susceptible to missing necessary motions. A large value for r_{tunnel} results in a distribution resembling uniformity or a distribution with an accumulation of samples at the boundaries of $\mathcal{X}_{free}$. This removes any advantage or even worsens HBFMT*'s performance compared with BFMT* or FMT*. Fig. 5 highlights the effect of different values for r_{tunnel} on the sampling distribution in $\mathcal{X}_{free}$.

Both r_{singe} and r_{tunnel} require tuning given robot and environment models. An experiment that varies r_{singe} and r_{tunnel} for planning problems in different environments is provided in Sec. 7 to empirically demonstrate the effect that these parameters have on algorithm performance.



(a) The MAMMOTH rover. (b) The MAMMOTH rover leg.

Figure 6: Labelled view of the MAMMOTH rover and its sub-components.

6 Planning for the MAMMOTH Rover

The RWMR used in all experiments is the MAMMOTH rover shown earlier in Fig. 1 and described in detail in (Reid et al., 2014). The physical MAMMOTH rover has a mass of 75 kg and 16 points of actuation. Its maximum footprint is 1500 mm by 1500 mm, while its most compact footprint is 650 mm by 650 mm. It may drive and steer each of its wheels independently and continuously allowing for omnidirectional driving. Each hip actuator can be moved between $\pm 270°$ to change the footprint of the vehicle. Lastly, each leg is a parallel structure. The legs can raise and lower between $-20°$ and $70°$ relative to the transverse plane of the rover's body.

A simulated model of MAMMOTH rover used for planning, shown in Fig. 7, has an 8-DOF configuration space. It may translate its body frame B relative to its inertial frame I along the x_B^I, y_B^I and z_B^I directions. Its roll, pitch and yaw positions are denoted by ϕ, θ and ψ respectively. In all experiments we hold ϕ and θ constant at $0°$, while ψ is free to rotate. Additionally, the four hip joints, q_{H1},...,q_{H4} may rotate, thereby changing the robot's contact footprint. A randomly sampled state $\underline{x}$ contains values for each of these eight degrees of freedom. The state $\underline{x}$ also contains four thigh joint variables, q_{U_1},..., q_{U_4} that are calculated as a function of the eight degrees of freedom as well as the local terrain profile. Additional dependent joints that are driven to meet a desired configuration are the ankle joints, $q_{S_1}, ..., q_{S_4}$ and wheel joints, $q_{A_1}, ..., q_{A_4}$. These joint positions and rates are determined as a function of the desired velocity of the platform. The MAMMOTH rover limb numbers and joint names are labelled in Figure 6.

6.1 Collision-Based Feedback Sampling

The 8-DOF configuration space does not characterize the capability of the MAMMOTH rover to raise one leg off the ground at a time. This ability is critical when the rover needs to clamber over obstacles. To allow this clambering behaviour to emerge, a sampling feedback method is used. If it is found that a sample from the 8-DOF space is deemed invalid due to leg i colliding with an obstacle, the sample is checked again with leg i raised. This is done by changing the leg's q_{U_i} value so that the leg is in its fully raised configuration. If this new leg-raised state is not in collision with obstacles and obeys static stability constraints, then it is deemed a valid sample $\mathcal{X}_{free}$. Alg. 8 provides pseudo-code for the sampling implementation used for the MAMMOTH rover planning problem. The new space being planned through has 9-DOF as there can only be one leg raised at a time.

By not uniformly sampling a configuration space that includes dimensions that describe the robot's ability to raise and lower its legs off the ground, asymptotic optimality and completeness guarantees are removed from the planner. To reduce this effect of feedback sampling, a sampling strategy that ensures that the 8-DOF $\mathcal{X}$ space is sampled uniformly for a minimum proportion, $(1-f)$, of the n samples is used. The correct way of ensuring these guarantees would be to uniformly sample the full 9-DOF space. As part of the collision detection procedure, the rover's static stability is checked according to the model described in (Papadopoulos and Rey, 1996). This characterizes static stability by finding the minimum angle between the force vector extending from the rover's centre of mass and the vector connecting each wheel/ground contact point. In all of the experiments performed to validate the motion planner a moderate stability margin of about 10° is used.

Algorithm 8: Sample Free Mammoth State With Collision Feedback

```
1  function SampleFree(n,Π,l,r_tunnel)
2      λ ← [ ]
3      for i = 1 to |Π| do
           if i mod 2 = 0 then
               λ ← [λ  Reverse(Π_i)]
           else
               λ ← [λ  Π_i]
4      m ← n
5      while m > 0 do
6          if m < (1 − l)n then
7              x ← SampleBiased(λ,r_tunnel)
8          else
9              x ← SampleUniform
10         q_U ← MammothIK(x)
11         if IsValid(x) then
12             m ← m − 1, S ← S ∪ x
           else
13             for i ← 1 to 4 do
14                 if LegCollision(i) then
15                     q_U(i) ← q_U.bounds.min
16                     if IsValid(x) then
17                         m ← m − 1, S ← S ∪ x
       return S
```

6.2 Cost Functions

For planning with HBFMT* two cost functions are required. The first is the cost function associated with the sub-space $\mathcal{X}^{\#}$, while the second is the cost associated with the full configuration space $\mathcal{X}$. In considering the MAMMTOH rover planning problem, state transition costs approximate the amount of mechanical work in a state transition. When planning in $\mathcal{X}^{\#}$, the cost function used is:

$$J^{\#}(\underline{x}_i, \underline{x}_{i+1}) \approx \mu M g \sqrt{(\Delta x_B^I)^2 + (\Delta y_B^I)^2} + g m_{body} \Delta z_B^I, \tag{3}$$

while in $\mathcal{X}$ the cost function is:

$$\begin{aligned} J(\underline{x}_i, \underline{x}_{i+1}) \approx \mu M g \sqrt{(\Delta x_B^I)^2 + (\Delta y_B^I)^2} \\ + g m_{body} \Delta z_B^I \\ + g m_{leg} (\Delta z_{leg,1}^I + \Delta z_{leg,2}^I + \Delta z_{leg,3}^I + \Delta z_{leg,4}^I) \\ + \mu m_{body} g r_{leg} \Delta\psi \\ + \mu m_{leg} g r_{leg} (|\Delta\psi - (q_{H_1,i+1} - q_{H_1,i})| + |\Delta\psi - (q_{H_2,i+1} - q_{H_2,i})| \\ + |\Delta\psi - (q_{H_3,i+1} - q_{H_3,i})| + |\Delta\psi - (q_{H_4,i+1} - q_{H_4,i})|). \end{aligned} \tag{4}$$

In each of equations, Δ is the absolute difference over a single dimension. Additionally, $m_{body} = 30$ kg is the mass of the rover's central chassis, $m_{leg} = 10$ kg is the mass of a single leg, $M = (m_{body} + 4m_{leg})$ is the total mass of the rover, $r_{leg} = 0.7$ m is the average reach of the center of mass of a single rover leg, $\mu = 0.1$ is an approximate rolling resistance coefficient for the rover's wheels, and $\Delta z_{leg,i}^I$ is the change in height of the centre of mass of limb i relative to the inertial frame. The constants used to generate the cost functions are an approximations; an item of future work is to incorporate a high fidelity physical model of the rover into the planner to more accurately calculate energy expenditure. The traversal cost of a path is the sum of its state transition costs:

$$c(\pi) = \sum_{i=0}^{b-1} J(\underline{x}_i, \underline{x}_{i+1}), \tag{5}$$

where b is the number of waypoints in the path.

7 Numerical Experiments

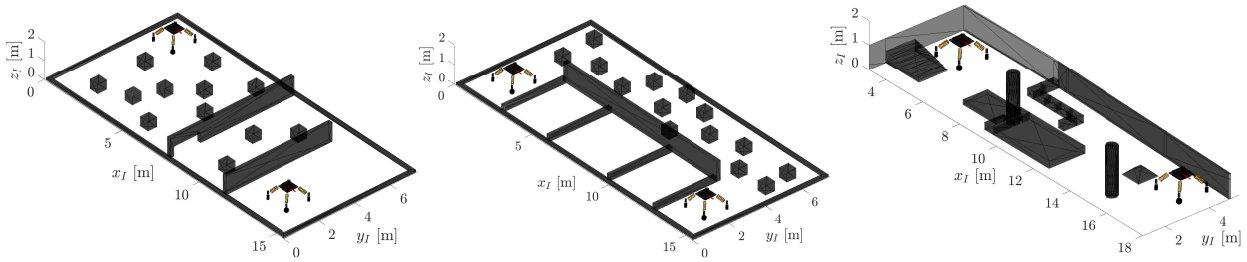
The aims of the numerical experiments are to evaluate HBFMT*'s performance against state-of-the-art sampling-based planners, including FMT*, BFMT*, RRT*, Informed RRT* and BIT*; demonstrate the algorithm's suitability to path planning for RWMRs using the system described in Sec. 6 and four distinct environments described in Sec. 7.1; and to evaluate the sensitivity of the hierarchical planning framework to parameter tuning.

Numerical experiments are performed using a variety of environments that are designed to highlight the advantages and disadvantages of HBFMT*. Planners are evaluated on the returned path cost, required computation time, as well as the probability of successfully finding a path at a certain computation time. All experiments are performed on a flat terrain that is cluttered with obstacles.

All planners are implemented with the Open Motion Planning Library (OMPL) version 1.1.0 (Sucan et al., 2012). OMPL implementations for FMT*, RRT*, Informed RRT* and BIT* have been used. The BFMT* and HBFMT* implementations are our own. Experiments are run on a 3.4 GHz Intel i7 processor with 32 GB of RAM running Linux Ubuntu 16.04 LTS. In all planners the Geometric Nearest-neighbor Access Tree (GNAT) algorithm (Gipson et al., 2013) is used to perform nearest neighbourhood queries, while the Flexible Collision Library (FCL) (Pan et al., 2012) is used to perform all collision checks.

7.1 Environments

Three separate environments are used to perform the experiments. The first two are purely simulated environments, however the third is based on an actual environment constructed within a Mars analogue environment. The three environments have the following designations: (a) Gauntlet, (b) Alternatives, and (c) Mars Lab Corridors. Each of the environments is shown in Fig. 7 along with the initial and goal states of the rover.



(a) Gauntlet. (b) Alternatives. (c) Mars Lab Corridors.

Figure 7: The three environments used in the numerical experiments along with the initial and goal states of the rover.

Table 1: The planners used in the numerical benchmarking experiment and whether or not the corresponding OMPL implementation was used. The numbers of samples per batch is given in MATLAB vector notation *<start number>:<increment number>:<end number>*.

Planner	OMPL	Trials/batch	Samples/batch
HBFMT*	No	50	1000:1000:5000, 5000:5000:25000
FMT*	Yes	50	1000:1000:5000, 5000:5000:25000
BFMT*	No	50	1000:1000:5000, 5000:5000:25000
BIT*	Yes	50	Anytime planner
RRT*	Yes	50	Anytime planner
Informed RRT*	Yes	50	Anytime planner

In the Gauntlet environment, the rover must navigate around/over box obstacles, through a tunnel and through a narrow passageway to get to its goal. This environment contains a single homotopy class of paths embedded within the Euclidean translational sub-space of the rover. HBFMT* is able to focus its search on this single family of paths. The Alternatives environment is designed to contain two homotopy classes of paths within the Euclidean translational sub-space. The robot may choose to clamber over the four step obstacles, or drive around and over the many block obstacles. Clambering over obstacles is energy intensive, so an optimal solution should lie in the region with the block obstacles. In order to successfully traverse the Mars Lab Corridors environment the robot must clamber over the set of step obstacles in the middle of the environment. A solution for this environment is only achievable if the planner considers the robot's ability to raise individual legs off the terrain. This is achieved by using the collision-based sampling feedback strategy described in Sec. 6.1.

7.2 Experiment Setup

For the benchmarking experiments, paths are found for the four environments using a set of sampling-based planners: HBFMT*, BFMT*, FMT*, BIT*, RRT* and Informed RRT*. A batch of trials is composed of 50 planner executions. For HBFMT*, BFMT* and FMT*, distinct batches are assigned between 1000 and 5000 nodes with 1000 node increments and between 5000 and 25000 nodes with 5000 node increments. HBFMT* additionally uses 1000 nodes to sample $\mathcal{X}^{\#}_{free}$. For BIT*, RRT* and Informed RRT*, single batches of 50 trials are used with an expiry time of 350 s. For RRT* and Informed RRT*, a goal-bias probability of 5% is used. This setup data is detailed in Table 1.

Table 2: The tuning parameters used for HBFMT* for each environment.

Environment	r_{singe} [m]	r_{tunnel} [kJ]	ℓ [-]
Gauntlet	2	0.1	0.2
Alternatives	3	0.75	0.2
Mars Lab Corridors	2	0.3	0.2

Tuning parameters for HBFMT* are different for each environment as detailed in Table 2. HBFMT* begins by sampling the $\mathcal{X}^{\#}$ space associated with the translational mechanical work of the rover. In this case the rover's body without legs is used for collision checking. Based on observations of best practice from (Janson et al., 2015), k-nearest neighbour versions of all of the planners are used. As discussed in Section 2 k-nearest neighbor search is more adaptable to a variety of obstacle configurations than its radial-nearest neighbor counterpart. The value k_n, which denotes the number of nearest neighbour nodes, is calculated as $k_{n,FMT*} = 2^d e/d \log(n)$ for the FMT* variants, and $k_{n,RRT*} = (e + e/d) \log(n)$ for RRT*. For HBFMT*, k_n is determined by changing the number of nodes n to $n\ell$ as in Eq. 2.

7.3 Results

All experimental results are presented in path cost versus time and success rate versus time plots. Results for the Gauntlet environment are shown in Fig. 8, Alternatives in Fig. 10 and Mars Lab Corridors in Fig. 11. In each of these figures, excluding Fig. 11 for Mars Lab Corridors, separate results are presented with no feedback sampling (NFS) and with feedback sampling (FS). Only results using feedback sampling are presented for Mars Lab Corridors given that leg lift manoeuvres are required to successfully traverse the environment. Additionally, the cost versus time plots in these figures are in a standard box plot format. Each box for the FMT* variant planners represents a distinct number of samples as described in Table 1. Similarly, in the success rate versus time plots, circles represent a distinct number of samples from the FMT* variant planners. A timelapse of a feasible solution from the Gauntlet environment is presented in Fig. 9. This example path illustrates fluid transition between intuitively discrete locomotion modes such as driving, clambering and footprint reconfiguration.

7.3.1 Gauntlet

From Figure 8, an obvious observation from the Gauntlet experiments is HBFMT*'s ability to find feasible solutions. For both the NFS and FS cases, HBFMT* returns more successful paths than any of the other planners. For the NFS case HBFMT* begins returning feasible paths 100% of the time in the 20,000 node batch. BFMT* and FMT* are the next most successful planners with an approximately 70% success rate by the 20,000 to 25,000 node batches. The anytime planners perform poorly in this environment for the NFS case with an approximately 5% success rate at the 350 s expiry time. Another significant improvement by the HBFMT* planner is the computation time when compared with BFMT* and FMT*. For 25,000 nodes the average computation time difference is 80 s with FMT* and 100 s with BFMT*.

When analyzing path cost results it must be noted that the reported cost data is only for the trials that successfully returned a path. For both the NFS and FS cases HBFMT* outperforms all other planners and begins to find successful solutions faster. For the NFS case there is an approximate cost percentage difference of 5% between the HBFMT* and FMT* variants between 120 s and 250 s. For the FS case on the other hand there is a larger 15% margin during the same time period. To explain this difference, it first must be noted that HBFMT* returns a similar cost in a similar timeframe in both the NFS and FS cases. BFMT* and FMT* have difficulty finding a path through the first narrow passageway shown in Fig. 9 in the NFS case. With FS, FMT* and BFMT* are able to use samples where the robot is straddling the low boundary walls. This results in a significant improvement in successfully returning paths, however due to energy intensive leg raising manoeuvres being utilized the average path cost increases. BIT* also has a notable increase in success rate when FS is enabled, however the cost of BIT*'s returned paths is 50% to 33% more expensive than the FMT* variants.

7.3.2 Alternatives

In the Alternatives environment HBFMT* may find paths through $\mathcal{X}^{\#}_{free}$ over the two homotopic regions. These regions include the area with the four steps and the area with block obstacles. The $\mathcal{X}^{\#}_{free}$ search of

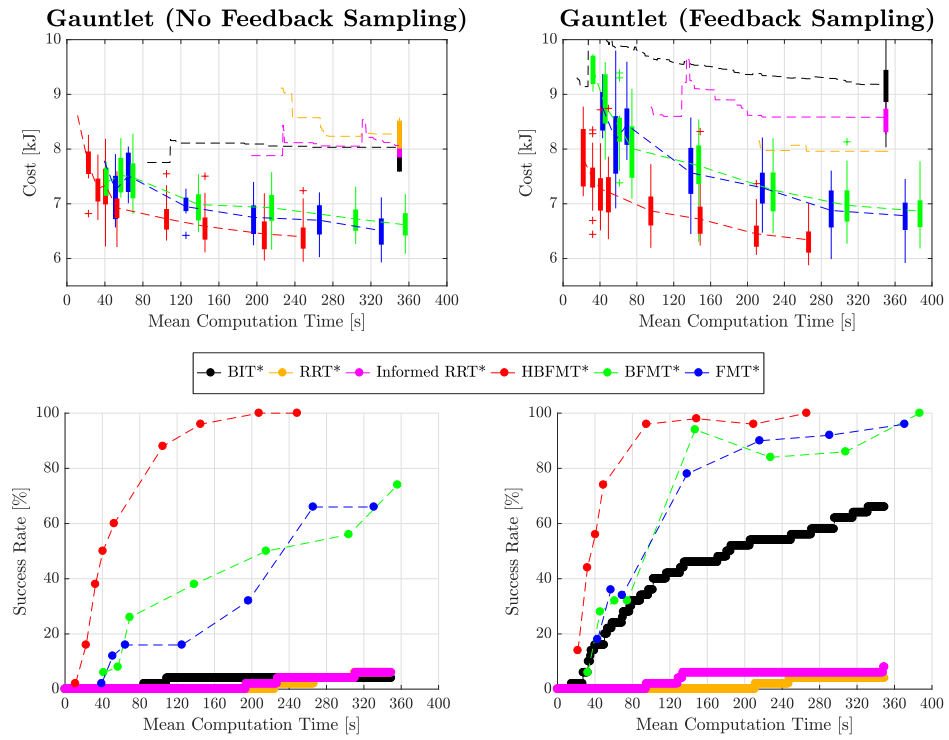


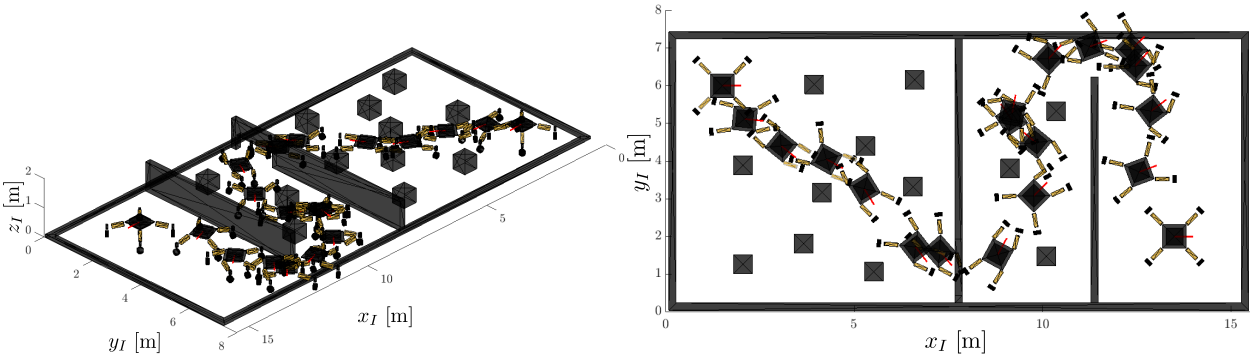
Figure 8: Experimental results from planners solving for a path through the Gauntlet environment. The left column shows results with no feedback sampling and the right column shows results with feedback sampling.

HBFMT* results in a biased sampling distribution in $\mathcal{X}_{free}$ that is focused over both distinct sections of the space. The cheaper cost region will lie in the region riddled with box obstacles. To traverse the area with four steps would require many energy intensive leg lifts. Additionally, finding a path over the step obstacles with feedback sampling is extremely unlikely given the four high-dimensional narrow passageways that are generated by the four required clambering manoeuvres.

Even though the number of samples available in the box obstacle region is diluted by sampling of the step region, a notable cost improvement between HBFMT* and the other planners is observed, especially for batches with smaller numbers of samples. Due to its bias around a sub-region of the enviornment's box region, HBFMT* converges to 100% successful solutions by 20 s; faster than any other planner when using NFS. BIT*, FMT* and BFMT* however are also relatively quick to converge to 100%, with BIT* reaching 100% convergence by 65 s, and BFMT*/FMT* by 100 s.

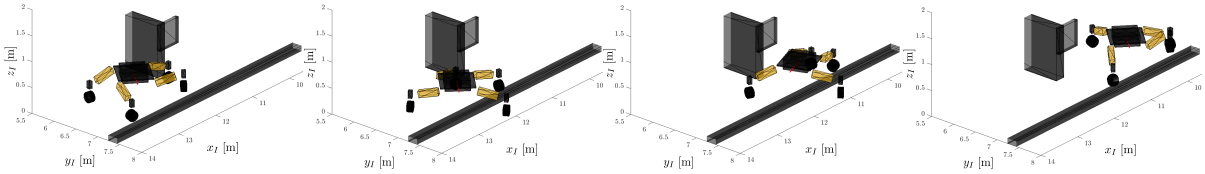
For the feedback sampling case the FMT* variants get close to but do not completely converge to 100% success rates. BIT* on the other hand does converge to 100% by 45 s. As mentioned in Sec. 6.1, due to the ninth 'leg-raise' dimension not being included in $\mathcal{X}_{free}$ and only being sampled if a leg collision occurs, AO and PC guarantees are removed. This effect is not noticeable with the anytime BIT* planner however given that it may resort to previously found solutions and not rely on a wavefront of edges expanding through a single batch of nodes.

HBFMT* once again consistently returns better cost paths faster than all of the other planners, especially at lower computation times (< 100 s). For the NFS case there is an average cost difference of 25% at 60 s and 14% at 180 s between HBFMT* and BFMT*/FMT*. For the FS case, the margins are similar with 27% difference at 60 s and 14% difference at 195 s.



(a) Isometric view.

(b) Top view.



(c) Leg 1 raises over the boundary wall.

(d) Straddling position with leg 1 lowered.

(e) Narrow passageway transition.

(f) Leg 1 raises over the boundary wall again.

Figure 9: A path generated by BFMT* using 25,000 nodes in which the MAMMOTH rover straddles the low boundary wall in the narrow passageway at $x \approx 11$ m and $y \approx 7$ m. The bottom row of images show the leg raising manoeuvres the MAMMOTH rover performs to clamber over the boundary wall.

7.3.3 Mars Lab Corridors

The Mars Lab Corridors environment forces the MAMMOTH rover to perform a clambering manoeuvre to traverse the 'S'-shaped obstacle at the centre of the space. For this reason, only the FS case is used in the Mars Lab Corridors experiment. The manoeuvre required for the robot to clamber over the obstacle represents a high-dimension narrow passageway where the robot must find a set of sequential moves to clamber over the obstacle while remaining stable. This narrow passageway sequence of manoeuvres includes one or more leg raises and yaw re-orientations. Bidirectional search is commonly used to address the narrow passageway problem and its efficacy is evident from the success versus computation time plot in Fig. 11. HBFMT* gets to 90% success by the 15,000 node batch, while BFMT* gets to just above 50% by 15,000 nodes. FMT* and the uni-directional anytime planners see success rates at or below 20% for all computation times. Due to its focussed exploration, HBFMT* is able to outperform BFMT* in terms of success rate.

If a solution is found, BFMT* and HBFMT* perform relatively similarly. If the clambering manoeuvre over the S-shaped obstacle is found, it is straight forward for any of the planners to achieve a solution that is close to the optimal given the absence of additional obstacles.

Generally, HBFMT* is observed to improve path quality with a smaller computation time. Notably, on average HBFMT* returns a greater number of lower-cost feasible solutions sooner than the other planners for all the environments presented.

7.4 Parameter Tuning

An experiment investigating the robustness of the algorithm given varying parameter values of singe radius r_{singe} and tunnel radius r_{tunnel} is performed. In this experiment the HBFMT* planner is run for the MAMMOTH rover in the Gauntlet, Alternatives and Mars Lab Corridors environments with the same start

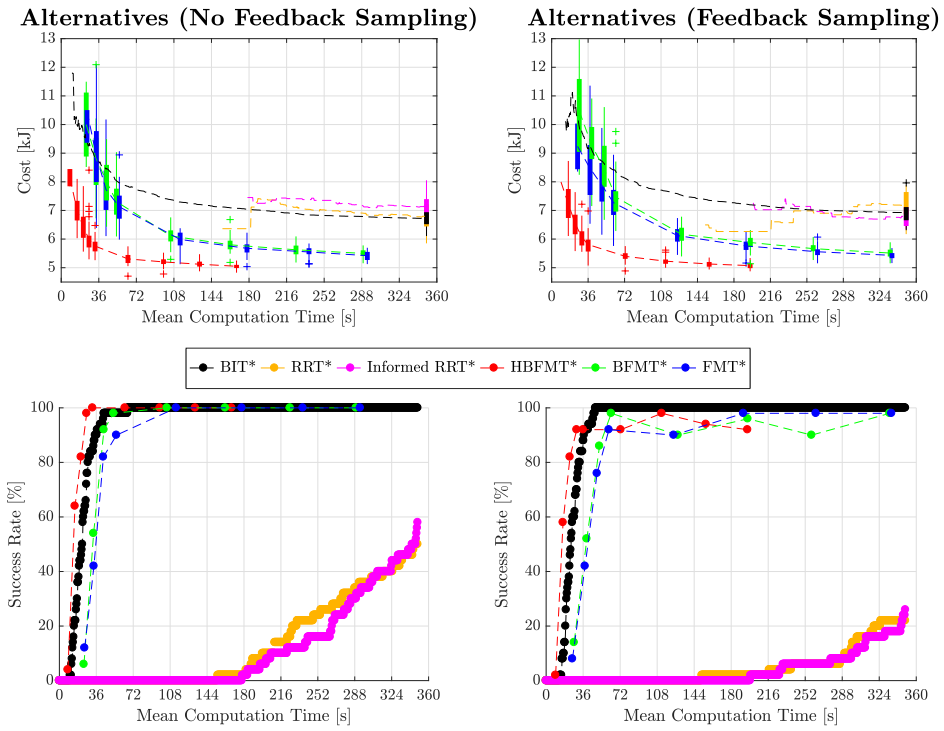


Figure 10: Experimental results from planners solving for a path through the Alternatives environment. The top row shows successful path cost versus time in standard box plot format. The left column shows results with no feedback sampling and the right column shows results with feedback sampling.

and goal configurations shown in Fig. 7. A total of 100 separate batches of trials are run, with 50 trials per batch. Each batch has a different r_{singe} and r_{tunnel} combination. The r_{singe} parameter is varied between 0.5 m and 5 m, while r_{tunnel} is varied between 0.1 kJ and 1.8 kJ.

The results of this experiment are shown in Fig. 12. Each mesh vertex represents the average of the 50 trials run for a specific r_{singe} and r_{tunnel} combination. The results for the Gauntlet environment demonstrate that lower cost is returned as the biased sampling gets more focused around the single $\mathcal{X}^{\#}_{free}$ homotopy class available in the environment. In Alternatives, it is seen that when r_{singe} is large and r_{tunnel} is small the number of successful traverses drops. This is caused by biased samples not being placed in the half of the environment filled with block obstacles and only being placed over the step obstacles. The planner has difficulty finding a path over the four step obstacles and therefore the success rate drops. Additionally, the Alternatives plot shows that if r_{tunnel} is too small then the biased sampling is too focused and cannot find configurations that successfully avoid the large number of block obstacles along the optimal path. A similar effect is seen with the Mars Lab Corridors environment.

8 Plan Following

The objective of the plan following trials are to (a) demonstrate the physical MAMMOTH rover following paths generated by the proposed HBFMT* planner; to (b) validate the cost function used in the planner against the actual energy expenditure by the physical platform; to (c) demonstrate actuator constraints (i.e. faults or sensor pointing restrictions) being incorporated into the plan; and to (d) highlight practical considerations of the MAMMOTH planning problem not originally considered in the numerical experiments. All successful experiment trials conducted may be viewed at https://goo.gl/5MC6Ib.

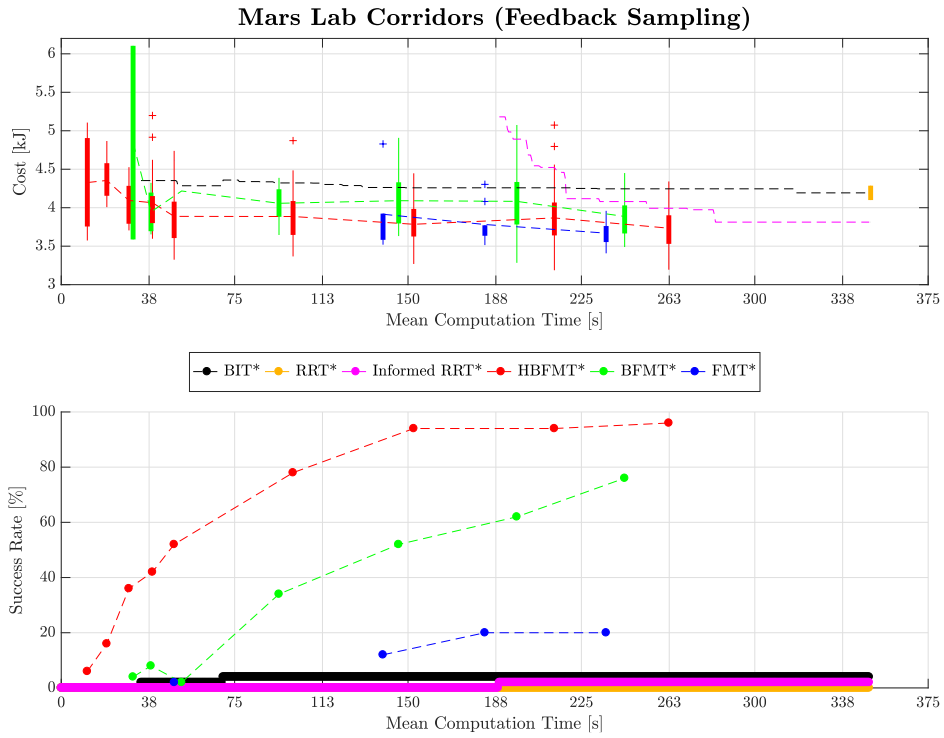
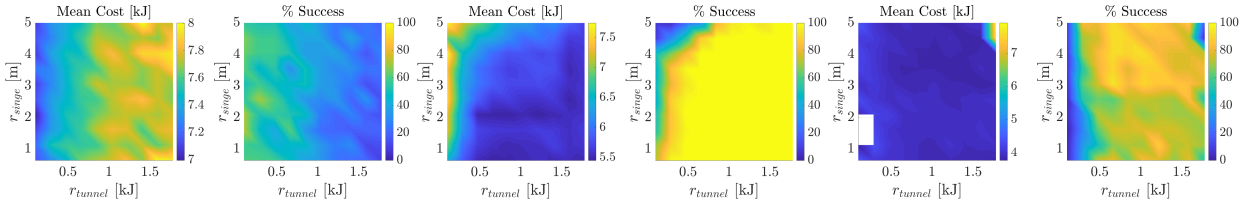


Figure 11: Experimental results from planners solving for a path through the Mars Lab Corridors environment using feedback sampling.



(a) Gauntlet (b) Alternatives (c) Mars Lab Corridors

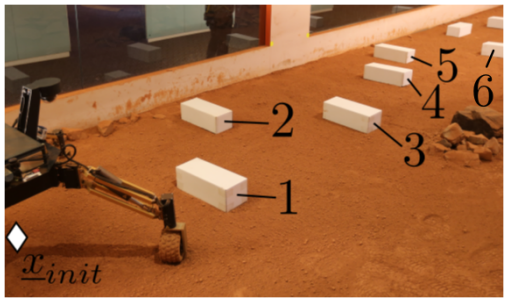
Figure 12: Path cost and success rates for HBFMT* with 5000 nodes in each environment with varying r_{singe} and r_{tunnel} values.

8.1 Experiment Setup

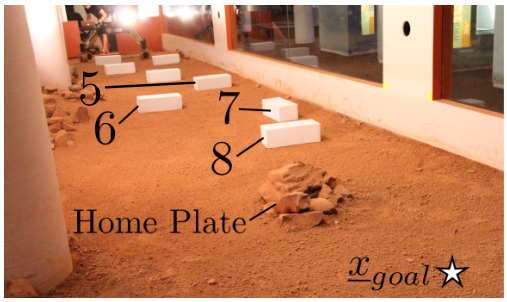
All experiments are performed at the Sydney Powerhouse Museum Mars Lab, a 20 m by 7 m space designed as a Mars analogue. The Mars Lab is a sandy terrain littered with rock obstacles.

The experiments are divided into trials where a single traverse of an environment is attempted. The execution time for a traverse is started as soon as motion in any actuator starts. Traverse execution time ends when the rover is within a tolerance ϵ of the final state. This ϵ was set as 0.05 kJ for all trials. All trials are performed by running the motion planner before the traverse is begun. The planner's generated path is given to the path follower which is used to control the motion of the rover. The rover drives along the path until it reaches its goal state.

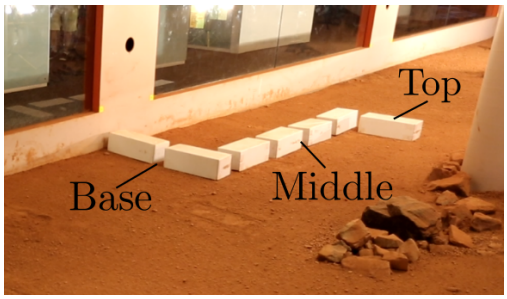
Trials are run for all environments using either n =5000 or n =10,000 samples in $\mathcal{X}_{free}$ and 5000 samples in $\mathcal{X}^{\#}_{free}$. Varying stability margins are used, while HBFMT* parameters are held constant over each trial at



(a) Mars Lab Boxes from near $\underline{x}_{init}$.



(b) Mars Lab Boxes from near $\underline{x}_{goal}$.



(c) Mars Lab Corridors from near $\underline{x}_{init}$.

(d) Mars Lab Corridors from near $\underline{x}_{goal}$.

Figure 13: The two environments used in the plan following experiment. Obstacles within the environment are cardboard boxes, rocks and boundary walls.

$r_{singe} = 0.2$ m, $r_{tunnel} = 0.2$ kJ and $\ell = 0.2$. Obstacle-based feedback sampling is enabled for all trials to allow the robot to raise legs over obstacles.

The path following experiments are run on two separate environments: *Mars Lab Boxes* and *Mars Lab Corridors*. The Mars Lab Boxes environment is a long thin wedge of the Mars Lab that is on a flat surface and is littered with box obstacles. The environment forces the robot to modify its footprint and possibly raise legs off the ground to get from one end of the space to the other. Each of the cardboard boxes in the environment is given an identifying number as shown in Fig. 13a and 13b. The Mars Lab Corridors environment is the same as the one described in Sec. 7.1 and shown in Figs. 13c and 13d. To traverse this environment, the rover must lift at least one leg over the 'S'-shaped obstacle in the centre of the space. The three sections of the 'S' are denoted as 'Base', 'Middle' and 'Top.' The environment is used to showcase how the planner accommodates for clambering manoeuvres using feedback sampling.

The initial and goal states for all trials are given in images of the two environments in Fig. 13. Obstacles within the environments consist of rocks and cardboard boxes. The planner has a map of the environment and the obstacles. All obstacles are inflated by a radius of 10 cm in the map. Obstacle inflation is needed due to uncertainty in the translational localization solution of the platform. Robot localization is performed using an array of fiducial markers mounted on the ceiling of the Mars Lab, an upwards facing camera and inertial measurement unit mounted on board the robot. The triangulated translational solution has an accuracy of ± 10 cm.

In selected trials sampling constraints are applied to various hip joint rotations or the body yaw rotation. These trials serve to highlight how the planner may be easily modified to accommodate for such cases as broken actuators or sensor pointing constraints. These trials are also performed to highlight how constraining the dimensionality of the explored configuration space may be beneficial in finding lower cost paths. To apply these constraints the sampling range of the associated constrained sub-space is reduced. Table 3 presents each of the successful path following trials executed by the MAMMOTH rover and the type of constraint(s)

applied during the trial.

Table 3: Setup details for each of the successful path traversals executed with the physical MAMMOTH rover.

Trial	Env.	n	β [°]	Constraint
10	Boxes	5,000	6	None
29	Boxes	5,000	23	None
47	Boxes	5,000	23	None
56	Boxes	10,000	23	None
55	Boxes	10,000	14	$q_{H_1} = 0°$
32	Boxes	10,000	14	$q_{H_1} = 0°, q_{H_3} = 0°$
49	Boxes	10,000	14	$q_{H_1} = 0°, q_{H_4} = 0°$
38	Boxes	10,000	23	$\psi = 0$
61	Corridors	5,000	14	$q_{H_1} = 0°, q_{H_4} = 0°$
66	Corridors	10,000	14	$q_{H_1} = 0°, q_{H_4} = 0°$

8.2 Experimental Results

A total of 66 trials are attempted, 10 of which are successful in traversing the pre-planned path completely without hitting any obstacles or experiencing other faults. Out of the remaining 46 trials, seven trials are 'partially successful,' in which the corner of an obstacle is hit but the robot's traversal is not impeded. A set of 21 trials are stopped due to a collision with an obstacle. The majority of obstacle collisions are caused by a noisy localization solution, while three are caused by obstacles not being included in the map. Tip-over stability failures occurred five times. These are caused by the use of too small a stability margin within the planner, or the introduction of an incline due to thigh linear actuator positions having different stroke values. Nine trials fail due to software faults, while the remaining two trials fail due to low battery charge on the emergency stop remote control system.

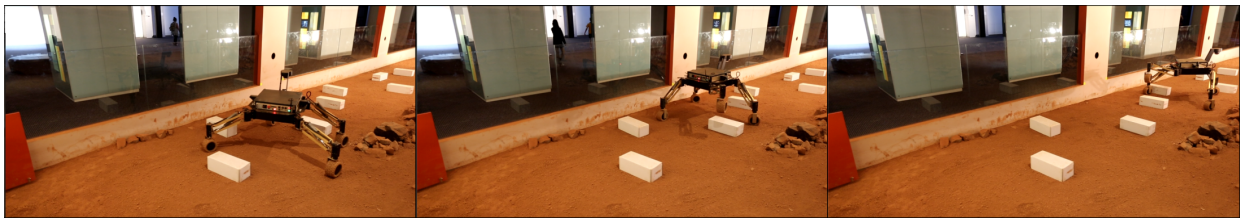
Time-lapse images taken from trials 56 and 61 are provided in Fig. 14 and Fig. 16 respectively. To demonstrate that the proposed planner can produce plans that can be followed by a real world reconfigurable rover, two separate trials (trials 56 and 61) from the two environments are presented in detail. Plots providing a detailed comparison between the physical and simulated trials are provided in Figs. 15 and 17.

For each trial a planned path is generated by HBFMT*, a simulation of the trial that uses the path following controller is run and then the physical traverse is attempted. Results from each of these steps are compared to validate the performance of the physical system in following a planned path. To validate the cost function used in planning with the physical system the root mean squared error (RMSE) and normalized root mean squared error (NRMSE) between each sub-space in each trial is found between the physical traverses and planned paths. The RMSE and NRMSE are calculated as:

$$RMSE(\hat{J}_i) = \sqrt{E\left[(\underline{\hat{J}}_i - \underline{J}_i)^2\right]}, \tag{6}$$

$$NRMSE(\hat{w}_i) = \frac{\sqrt{E\left[(\underline{\hat{J}}_i - \underline{J}_i)^2\right]}}{\hat{J}_{i_{max}} - \hat{J}_{i_{min}}} \tag{7}$$

where $\hat{J}_i$ is the vector of accumulated mechanical work costs for a sub-space i in the physical trial, while J_i is the vector of accumulated mechanical work costs for sub-space i in the planned path. The RMSE and NRMSE results for each trial are presented in Fig. 18. Given that the traverse duration may be different



(a) $t = 45$ s, $\tau = 0.14$ (b) $t = 89$ s, $\tau = 0.29$ (c) $t = 134$ s, $\tau = 0.43$

(d) $t = 179$ s, $\tau = 0.57$ (e) $t = 224$ s, $\tau = 0.71$ (f) $t = 268$ s, $\tau = 0.86$

Figure 14: Timelapse of trial 56 with a time duration of 313 s. Images **(a-c)** show the traverse from a point of view close to the initial state, while images **(d-f)** show the traverse from a point of view close to the goal state. A video of the full trial is available at https://youtu.be/fZ7X2Nz8lhA.

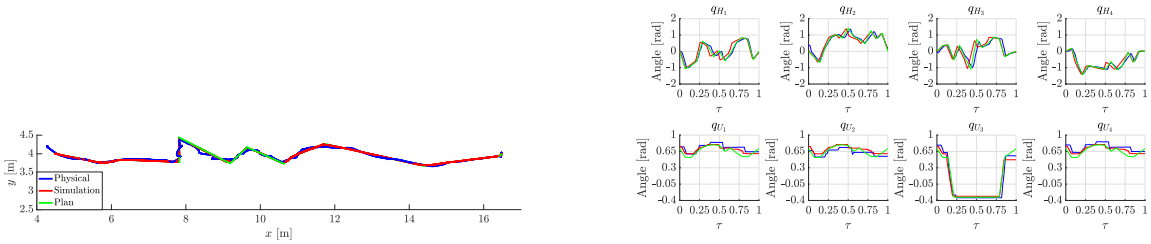
between the plan, simulation and physical traverse, a normalized traverse duration measure $\tau \in [0, 1]$ is used to compare the three. The τ parameter is the proportion of traverse executed so far.

8.2.1 Trial 56

This trial uses HBFMT* to plan a path through the Mars Lab Boxes environment. Initial settings for the planner include a sub-space batch size $n^{\#} = 5000$, a full space batch size $n = 10,000$, a minimum stability margin $\beta = 23°$ and no motion constraints. Additionally, the tunable HBFMT* parameter values are set to $r_{singe} = 2$ m, $r_{tunnel} = 0.2$ kJ and ℓ =0.2. Upon completion, the planner found a path with a mechanical work cost of 4.65 kJ with a computation time of 206 s. In its sub-space search two separate solutions are found. The simulated MAMMOTH plan results in a mechanical work cost of 4.71 kJ, while the physical experiment resulted in a mechanical work cost of 5.07 kJ.

The trial 56 traverse is now described; a video of the trial can be found at https://youtu.be/fZ7X2Nz8lhA, while a time-lapse is shown in Fig. 14. The rover begins begins its traverse by driving directly forward. At 0:12 it begins to transition into a stable tripod state with legs 1, 2 and 4 as the contact points of the tripod in preparation for an impending leg 3 lift. At the same time the rover re-orients its yaw to prevent leg 1 from hitting box 3. At 0:40 leg 3 begins to raise off the ground, initially to clear box 3. Excessive steering motion is observed at 1:24 due to noisy localization in this region of the yard. From 1:52 to 2:32 the rover yaws by over 90° to get leg 2 out of the path of box 5 and prepare leg 4 to avoid box 6. From 2:32 to 3:27 the rover continues to yaw by 180° so that it is almost facing backwards allowing leg 4 to drive past box 7 on the box's wall side. From 3:27 onwards the rover drives directly towards the xy position of $\underline{x}_{goal}$ as it is already in a satisfactory configuration to traverse over the last rock obstacle. As the rover traverses over 'Home Plate' it lowers leg 3. Lastly, it performs a yaw manoeuvre of $-90°$ to reach the final state. It is observed that leg 1's steering joint slightly scuffs the wall during this manoeuvre. This minor collision may be attributed to an incorrect wall location within the map.

A notable aspect of this path is how the robot raises a single leg for the majority of the traverse. It is observed in Fig. 19 that there is a smaller mechanical work cost accumulated by the hip joints than in trials 10, 29 and 47, the other unconstrained Mars Lab Boxes trials. The trial 56 plan trades hip motion for increased yaw motion compared to trials 10, 29 and 47. It is also observed from this plot that the q_{U_3}



(a) Body position.

(b) Actuator position.

Figure 15: Trial 56 plots comparing workspace position and actuator positions between the physical trials (blue), simulated trials (red) and planned paths (green).

contribution to the total mechanical work cost is almost four times the cost of any of the other thigh joints in trial 56. This is due to the leg raise and lower manoeuvres performed by leg 3 and the relatively small displacements performed by the other legs' thighs.

In Fig. 15a the x_B^I and y_B^I position of the rover in the physical traverse is compared with the planned path and simulated traverse. It is noted that the reported position in the physical experiment is more jagged than that of the plan and simulation. This may be attributed to noise in the localization solution provided by the fiducial localization system. It is also observed that in the physical trial the robot starts moving slightly behind the initial position used in the simulated trial.

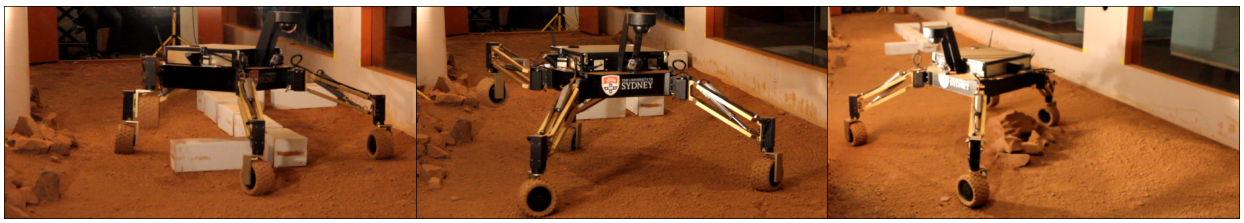
In Fig. 15b the physical hip motions are seen to track the simulated trial, however an offset is present in part due to the physical trial's initial position offset. The hip tracking performance is reiterated in Fig. 18 where RMSE values for each hip cost were are below 50 J and NRMSE values below 0.2. The thigh joints do not have as effective tracking performance as shown in Figs. 15b. This is attributed to the limited velocity control on each of the linear actuators. The NRMSE values for the hip actuators are generally multiple times higher than the hip NRMSE values for this reason, as shown in Fig. 18a. For the physical trials the linear actuators are driven at a constant speed of 11 mm/s to make it easier to synchronize all thighs.

8.2.2 Trial 61

Trial 61 uses HBFMT* to plan a path through the Mars Lab Corridors environment. The traverse uses a batch of $n^{\#} = 5000$ samples for subspace exploration, a batch of $n = 10{,}000$ samples for the full space exploration and a tip-over stability constraint of $\beta = 14.3°$. Two angular constraints are applied, with $q_{H_1} = 0°$ and $q_{H_4} = 0°$. HBFMT* is used with the parameter values: $\ell = 0.2$, $r_{tunnel} = 0.2$ and $r_{singe} = 2.0$. The plan is generated in 136.5 s with a mechanical work cost of 4.4 kJ. Two alternative paths are found through the sub-space. The simulation result uses 4.2 kJ while the physical trial uses 4.39 kJ in mechanical work cost.

A video of trial 61 may be viewed at https://youtu.be/2vwEWn_SwGE. Like in trial 56 the rover begins moving slightly behind the desired initial state. Its first set of motions move the rover to the desired initial state by driving the whole platform forward and to the right, while also slightly raising the platform. The rover then begins to drive through open space towards the 'S'-shaped obstacle while also performing a slight counter-clockwise yaw. At 0:12 the rover begins to rotate hips 2 and 3 in preparation for a tripod configuration with leg 1 lifted. At 0:33 leg 1 is lifted so that the rover can move it over the base of the 'S'-shaped obstacle. At 0:54 leg 1 is being moved over the obstacle while hip 2 rotates back towards $q_{H_2} = 0°$. At 1:18 leg 1 begins to lower and a static instability is approached as hip 2 continues to be rotated. At 1:34 leg 1 has not lowered fast enough and the robot becomes unstable and falls onto leg 1. While falling, leg 1 continues to lower, however it does not reach the same level as the other thighs leaving the robot with a slight pitch and roll. At 2:06 the rover begins to yaw in a counter-clockwise direction so that legs 2 and

(a) $t = 45$ s, $\tau = 0.14$ (b) $t = 91$ s, $\tau = 0.29$ (c) $t = 136$ s, $\tau = 0.43$



(d) $t = 182$ s, $\tau = 0.57$ (e) $t = 227$ s, $\tau = 0.71$ (f) $t = 273$ s, $\tau = 0.86$

Figure 16: Timelapse of the trial 61 traverse with a time duration of 318 s. Images **(a-c)** show the traverse from a point of view close to the initial state, while images **(d-f)** show the traverse from a point of view close to the goal state. A video of the full trial is available at https://youtu.be/2vwEWn_SwGE.

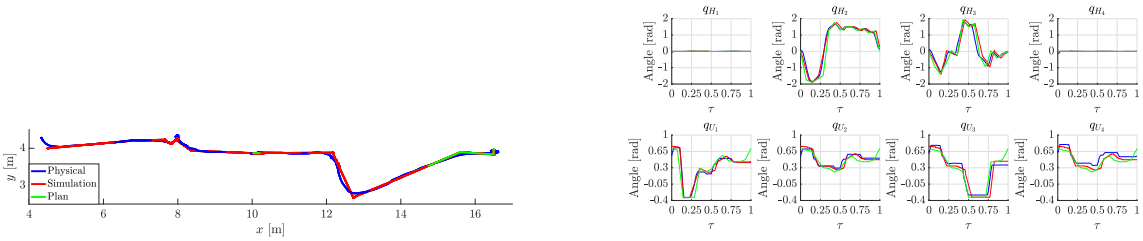
3 will pass around the base of the 'S'-shaped obstacle. While this is happening, hips 2 and 3 rotate in a positive direction to form a tripod stance with leg 1, preparing for a lift of leg 3. At 2:22 leg 3 is lifted so that it does not hit the top of the obstacle. At 2:41 leg 4 passes through the narrow passageway, and at 2:46 the platform yaws positively to bring leg 2 through the narrow passageway. At 3:29 the platform begins to yaw in the clockwise direction to prepare for the traverse over the final rock obstacle. At 4:28 the platform begins to pass over the final rock obstacle straddling the rock initially with legs 1 and 2. By 4:52 the rock obstacle has been traversed and there is a yaw motion back to the goal state.

This traverse demonstrates the robot's ability to clamber over an obstacle. Additionally, it demonstrates the ability to impose actuator constraints within the planner. The clambering manoeuvre is planned for by using the feedback sampling method described in Sec. 6.1. Given that the two front hips are constrained the robot must rely on the two back hips for reconfiguration and ensuring that static stability is maintained. The first leg raise manoeuvre lifts leg 1 over the 'S'-shaped obstacle. To do this it is critical that the hip 2 swing around to a negative value to ensure static stability. It is noted that thigh 1 does not lower fast enough when leg 1 is being lowered back into ground contact to prevent an unstable state from being reached and the rover tipping onto leg 1. This is caused by the path following controller not issuing a thigh actuation value until it is too late. The lack of controllability of the thigh actuation controller, as discussed in Trial 56, is the root cause of this problem. Even so, the rover's traverse is not detrimentally affected by this tip-over with the robot simply tipping into a leg 1 contact state.

The RMSE values for mechanical work cost in Trial 61 shown in Fig. 18a demonstrate that the planned paths and physical trials return relatively similar results. The highest NRSME values are seen in the two constrained sub-spaces q_{H_1} and q_{H_4} as seen in Trial 66. This behaviour is due to the small maximum range in the denominator when calculating NRMSE for a constrained sub-space.

8.2.3 Planner Validation

This section analyzes how well the planned paths and physical traversals agree with each other. In doing so the planner's ability to find a path for a physical reconfigurable system is validated. Furthermore, the cost function used by the planner is compared with the actual energy consumption of the rover during a traversal.



(a) Body position.

(b) Actuator position.

Figure 17: Trial 61 plots comparing workspace position and actuator positions between the physical trials (blue), simulated trials (red) and planned paths (green).

Mechanical work costs calculated using Eq. (4) are presented in Fig. 19 for each of the physical trials and planned paths, while Fig. 18 shows the RMSE and NRMSE between the mechanical work costs for each sub-space for planned paths and physically followed paths in each trial.

In Fig. 19, the percentages of total cost attributed to each sub-space are visualized in each bar division. Every odd bar with a 'P' at the beginning of its trial name is associated with a physical trial, while every even bar that has a 'S' at the beginning of its trial name is associate with a planned path. It is observed in each of these plots that the mechanical work due to translation of the robot is the motion that dominates total cost, with over 50 % of the cost in each trial. The next most dominant mobility mode is the body yaw ψ, except for trial 4, which had its yaw constrained to $\psi = 0°$. There is a general tendency across all trials, except for trial 61, for the translation mechanical work in the physical trials to be greater than the planned paths. This may be attributed to noise in the physical localization solution, as shown in Fig. 15a. This is especially noticeable in trial 47, where the discrepancy between physical and simulated trials is about 0.75 kJ.

For the most part, agreement between mechanical work cost in the simulated traversals and the physical traversals has been demonstrated, however the actual energy expenditure is not given by this cost function. The rover's true energy expenditure is calculated as

$$E = \sum_{i=0}^{m} \sum_{t_j=1}^{t_f} I_i(t_j) V_i(t_j)(t_j - t_{j-1}) \tag{8}$$

where E is the total traversal energy expenditure in Joules, $I_i(t_j)$ is the current drawn by actuator i at time t_j, $V_i(t_j)$ is the operating voltage of actuator i at time t_j. Fig. 20 presents the actual energy expenditure by the MAMMOTH rover during each successful traverse.

There is an obvious discrepancy between the actual energy costs and the mechanical work costs, with the actual energy costs being an order of magnitude larger. The first discrepancy is seen with the introduction of the steering actuators $\underline{q}_S$. These actuators are responsible for changing the steering angle of each of the wheels and are not considered in the mechanical work cost function as their displacement is a function of the velocity of the platform, which is not considered by the planner. The energy required to move these actuators is dependent on the loading on a leg as well as the coefficient of friction between the wheel contact ellipse and the ground. The steering joint angle is a function of the velocity of the body frame as well as the angular velocities of the associated leg joints. An item of future work would be to make the planner kinodynamic and incorporate the velocity state of the vehicle. From Fig. 20 it is observed that the total steering actuator energy cost is similar to that of the total hip joint energy cost and larger than the total wheel drive energy cost. Energy cost for steering would be reduced by introducing smooth joint velocity transitions as well as incorporation of the steering cost into the planner.

From Fig. 20 it can be seen that the energy cost accrued by the hip actuators is significant and in the majority of trials the most costly sub-spaces. The energy required for the hip actuators does not reflect costs only proportional to displacement, as is obvious in the bar for trial 49 where q_{H_1} and q_{H_4} were constrained yet still require significant amounts of energy to hold their respective positions over the traverse. The energy required for holding position is not considered in the planner's cost function and is an obvious contributor to total energy cost. It is also noted that the energy cost due to the thigh joints $\underline{q}_U$ is marginal compared to the other actuated joints. This is due to the non back-drivable linear thigh actuators that do not require power to hold a position. Given this hardware design feature the energy cost of operating the linear actuators is due to motion of the actuators and not due to holding forces. For future statically stable reconfigurable systems it may be desirable to use non back-drivable or braked actuators to ensure that energy is not wasted in holding a configuration.

9 Discussion and Future Work

The aim of this work is to develop an autonomous planner used by a real-world RWMR system to find paths over unstructured terrain that leverage its many degrees of freedom. This aim has been realized, which is an important step in making the case for the use of RWMRs in applications such as planetary exploration, forestry, mining and search and rescue.

A significant result of this work is the generation of planned paths that utilize versatile and fluid RWMR locomotion, which to a human operator may be unintuitive. By structuring the planning task in a hierarchical manner, the intuitive sub-dimensional structure of the problem is quickly explored to then allow for a more focussed search of high-dimensional spaces to uncover these unintuitive behaviours. This work demonstrates some of the first physical planning trials for a robot that uses its reconfigurability for locomotion as well as some of the first physical realizations of hierarchical planning for reconfigurable robotic platforms.

The proposed HBFMT* planner is amongst the first hierarchical planners to use continuous domain sampling-based planning at all hierarchical levels, thereby making the planner both simpler to implement and immune to the configuration space discretization pitfalls of grid-based planners. The planner takes advantage of FMT* with biased sampling asymptotic optimality guarantees presented in (Janson et al., 2015) to retain AO. Bidirectional search is utilized to speed up computation time at the low-level of the hierarchy and to find multiple candidate paths in the sub-dimensional state space explored at the high level. Numerical results demonstrate that HBFMT* returns more feasible lower cost paths in a shorter amount of time compared with other AO planners such as the FMT* planners, BIT*, RRT* and Informed-RRT*.

The demonstration of HBFMT* sampling-based planning working with real robotic hardware is one of the first demonstrations of the use of AO sampling-based planning for global full body planning of a physical RWMR. Moreover, the planner validation experiments demonstrate novel capabilities such as clambering via feedback sampling and accounting for 'broken' actuators or sensor pointing constraints by applying motion constraints to sampling routines.

This work has attempted to tie together state-of-the-art planning theory with a real world system to both validate the theory and to expose shortcomings in the application of the theory to the physical robot. Naturally, future work focusses on addressing these shortcomings. Regarding future RWMR designs, it is observed that the energy cost of the non back-drivable linear actuators in each of the MAMMOTH rover thigh joints was minimal compared to the other joints that had to expend energy to hold their positions. For future RWMR designs, it would be highly desirable to use similar non back-drivable actuators at the hip joints, which were observed to consume the majority of energy during each traverse even when their motion was constrained.

The use of sub-dimensional decomposition to bias sampling has obvious performance effects. It is evident in (Vernaza and Lee, 2012) and (Rowekamper et al., 2013) that learning sub-dimensional decomposition

results in increased planning efficiency. The HBFMT* algorithm is flexible in that more levels may be added into the planning hierarchy besides just planning in the workspace and full state space. A future step in the development of this algorithm would be to integrate a learned intermediate sub-dimensional space. This would most likely take the form of planning over sets of motion primitives or task space regions.

Multiple implementation improvements could be made to make HBFMT* more efficient in using information from its sub-space exploration to inform the full space exploration. One such implementation improvement could be to proportionally scale the r_{tunnel} parameter along $\Pi^{\#}$ as a function of local $\mathcal{X}^{\#}$ sample density. Such a modification would help inform the full state space planner about local regions such as narrow passageways or densely packed obstacles where more focused sampling would assist in finding valid robot configurations. Another improvement could be to assemble $\Pi^{\#}$ from the best paths returned from multiple explorations using different cost functions. Such cost functions could be a bottleneck cost that preferences obstacle clearance (Solovey and Kleinbort, 2018) or a weighted Euclidean metric.

A lesson learned from the plan following experiments is that a high-fidelity energy model of the robot is required to better capture the predicted energy expenditure of the rover. Usage of a fast high fidelity dynamic model to evaluate candidate traversals between waypoints may help address this problem. Similarly, the non-holonomic ankle joints were not accounted for in the HBFMT* planner state space formulation. The planning examples presented in this work considered the geometric planning problem and it would be an obvious extension of this work to formulate a kinodynamic version of the planning problem.

Acknowledgements

This work was supported by the Australian Centre for Field Robotics; the NSW Government; and the Faculty of Engineering and Information Technologies, The University of Sydney, under the Faculty Research Cluster Program. We would like to thank Muhammad Esa Attia and Javier Martinez for assisting in the design and operation of the MAMMOTH rover.

References

Arslan, O. and Tsiotras, P. (2013). Use of relaxation methods in sampling-based algorithms for optimal motion planning. In *Proc. of IEEE ICRA*, pages 2421–2428.

Arvidson, R., Bell, J., Bellutta, P., Cabrol, N., et al. (2010). Spirit Mars Rover Mission: Overview and selected results from the northern Home Plate Winter Haven to the side of Scamander crater. *J Geophys. Res.-Planet*, 115(E7).

Arvidson, R. E., Iagnemma, K. D., Maimone, M., Fraeman, A. A., et al. (2017). Mars Science Laboratory Curiosity rover megaripple crossings up to sol 710 in Gale crater. *J. Field Robot.*, 34(3):495–518.

Arvidson, R. E., Squyres, S. W., Anderson, R. C., Bell, J. F., et al. (2006). Overview of the Spirit Mars Exploration Rover mission to Gusev Crater: landing site to Backstay Rock in the Columbia Hills. *J. Geophys. Res.*, 111(2).

Belter, D., Labecki, P., and Skrzypczynski, P. (2015). Adaptive motion planning for autonomous rough terrain traversal with a walking robot. *J. Field Robot.*, 33(3):337–370.

Bohlin, R. and Kavraki, L. E. (2000). Path planning using lazy PRM. In *Proc. of IEEE ICRA*, pages 521–528.

Brock, O. and Kavraki, L. (2001). Decomposition-based motion planning: A framework for real-time motion planning in high-dimensional configuration spaces. In *Proc. of IEEE ICRA*, pages 1469–1474.

Brunner, M., Fiolka, T., Schulz, D., and Schlick, C. M. (2015). Design and comparative evaluation of an iterative contact point estimation method for static stability estimation of mobile actively reconfigurable robots. *Robot. Auton. Syst.*, 63:89–107.

Cordes, F., Kirchner, F., and Babu, A. (2018). Design and field testing of a rover with an actively articulated suspension system in a Mars analog terrain. *J. Field Robot.*

Dobson, A. and Bekris, K. E. (2014). Sparse roadmap spanners for asymptotically near-optimal motion planning. *Int. J. Robot. Res.*, 33(1):18–47.

Esposito, J. M. and Wright, J. N. (2016). Matrix completion as a post-processing technique for probabilistic roadmaps. In *Proc. of WAFR*.

Ferguson, D. and Stentz, A. (2006). Anytime RRTs. In *Proc. of IEEE/RSJ IROS*, pages 5369–5375.

Gammell, J. D. and Barfoot, T. D. (2014). The Probability Density Function of a Transformation-based Hyperellipsoid Sampling Technique. Technical report, University of Toronto, TR-2014-JDG004.

Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. (2014). Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *Proc. of IEEE/RSJ IROS*, pages 2997–3004.

Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. (2015). Batch Informed Trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *Proc. of IEEE ICRA*, pages 3067–3074.

Gipson, B., Moll, M., and Kavraki, L. E. (2013). Resolution independent density estimation for motion planning in high-dimensional spaces. In *Proc. of IEEE ICRA*, pages 2437–2443.

Grotzinger, J. P., Sumner, D. Y., Kah, L., Stack, K., et al. (2014). A habitable fluvio-lacustrine environment at Yellowknife Bay, Gale Crater, Mars. *Science*, 343(6169).

Hauser, K. (2015). Lazy collision checking in asymptotically-optimal motion planning. In *Proc. of IEEE ICRA*, pages 2951–2957.

Hauser, K., Bretl, T., Latombe, J.-C., Harada, K., and Wilcox, B. (2008). Motion planning for legged robots on varied terrain. *Int. J. Robot. Res.*, 27(11-12):1325–1349.

Hauser, K. and Zhou, Y. (2016). Asymptotically optimal planning by feasible kinodynamic planning in a state–cost space. *IEEE T. Robot.*, 32(6):1431–1443.

Hsu, D. (2000). *Randomized Single-query Motion Planning in Expansive Spaces*. PhD thesis, Stanford University.

Ichter, B., Harrison, J., and Pavone, M. (2018). Learning sampling distributions for robot motion planning. In *IEEE ICRA*, pages 7087–7094.

Janson, L., Ichter, B., and Pavone, M. (2018). Deterministic sampling-based motion planning: Optimality, complexity, and performance. *Int. J. Robot. Res.*

Janson, L., Schmerling, E., Clark, A., and Pavone, M. (2015). Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *Int. J. Robot. Res.*, 34(7):883–921.

Jordan, M. and Perez, A. (2013). Optimal bidirectional rapidly-exploring random trees. Technical report, MIT CSAIL, MIT-CSAIL-TR-2013-021.

Karaman, S. and Frazzoli, E. (2010). Optimal kinodynamic motion planning using incremental sampling-based methods. In *IEEE CDC*, pages 7681–7687.

Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.*, 30(7):846–894.

Karumanchi, S., Edelberg, K., Baldwin, I., Nash, J., et al. (2016). Team RoboSimian: Semi-autonomous mobile manipulation at the 2015 DARPA Robotics Challenge finals. *J. Field Robot.*, 34(2):305–332.

Kavraki, L. E., Svestka, P., Latombe, J.-C., and Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robotic. Autom.*, 12(4).

LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press, 1 edition.

LaValle, S. M. and Kuffner, J. J. (2001). Randomized kinodynamic planning. *Int. J. Robot. Res.*, 20(5):378–400.

Li, Y., Littlefield, Z., and Bekris, K. E. (2016). Asymptotically optimal sampling-based kinodynamic planning. *Int. J. Robot. Res.*, 35(5):528–564.

Otte, M. and Correll, N. (2013). C-FOREST: Parallel shortest path planning with superlinear speedup. *IEEE Trans. Robot.*, 29(3):798–806.

Pan, J., Chitta, S., and Manocha, D. (2012). FCL: A general purpose library for collision and proximity queries. In *Proc. of IEEE ICRA*, pages 3859–3866.

Papadopoulos, E. G. and Rey, D. A. (1996). A new measure of tipover stability margin for mobile manipulators. In *Proc. of IEEE ICRA*, pages 3111–3116.

Plaku, E., Kavraki, L. E., and Vardi, M. Y. (2010). Motion planning with dynamics by a synergistic combination of layers of planning. *IEEE Trans. Robot.*, 26(3):469–482.

Reid, W., Fitch, R., Göktoğan, A. H., and Sukkarieh, S. (2016a). Motion planning for reconfigurable mobile robots using hierarchical fast marching trees. In *Proc. of WAFR*.

Reid, W., Göktoğan, A. H., and Sukkarieh, S. (2014). Moving MAMMOTH: Stable motion for a reconfigurable wheel-on-leg rover. In *Proc. of ARAA ACRA*.

Reid, W., Pérez-Grau, F. J., Göktoğan, A. H., and Sukkarieh, S. (2016b). Actively articulated suspension for a wheel-on-leg rover operating on a Martian analogue surface. In *Proc. of IEEE ICRA*, pages 5596–5602.

Rickert, M., Brock, O., and Knoll, A. (2014). Balancing exploration and exploitation in motion planning. *IEEE Trans. Robot.*, 30(6):2812–2817.

Rowekamper, J., Tipaldi, G. D., and Burgard, W. (2013). Learning to guide random tree planners in high dimensional spaces. In *Proc. of IEEE/RSJ IROS*, pages 1752–1757.

Salzman, O. and Halperin, D. (2015). Asymptotically-optimal motion planning using lower bounds on cost. In *Proc. of IEEE ICRA*, pages 4167–4172.

Salzman, O., Shaharabani, D., Agarwal, P. K., and Halperin, D. (2013). Sparsification of motion-planning roadmaps by edge contraction. *Int. J. Robot. Res.*, 33(14):4098–4105.

Sanchez, G. and Latombe, J.-C. (2003). A single-query bi-directional probabilistic roadmap planner with lazy collision checking. In *Proc. of ISRR*, pages 403–417.

Satzinger, B. W., Lau, C., Byl, M., and Byl, K. (2015). Tractable locomotion planning for RoboSimian. *Int. J. Robot. Res.*, 34(13):1541–1558.

Schmerling, E., Janson, L., and Pavone, M. (2015a). Optimal sampling-based motion planning under differential constraints: the drift case with linear affine dynamics. In *IEEE CDC*, pages 2574–2581.

Schmerling, E., Janson, L., and Pavone, M. (2015b). Optimal sampling-based motion planning under differential constraints: The driftless case. In *Proc. of IEEE ICRA*, pages 2368–2375.

Solovey, K. and Kleinbort, M. (2018). The critical radius in sampling-based motion planning. In *RSS*.

Starek, J. A., Açıkmeşe, B., Nesnas, I. A., and Pavone, M. (2016). Spacecraft autonomy challenges for next-generation space missions. In *Advances in Control System Technology for Aerospace Applications*, pages 1–48. Springer.

Starek, J. A., Schmerling, E., Janson, L., and Pavone, M. (2014). Bidirectional fast marching trees: An optimal sampling-based algorithm for bidirectional motion planning. In *Proc. of WAFR*.

Sucan, I., Moll, M., and Kavraki, L. E. (2012). The open motion planning library. *IEEE Robot. Autom. Mag.*, pages 72–82.

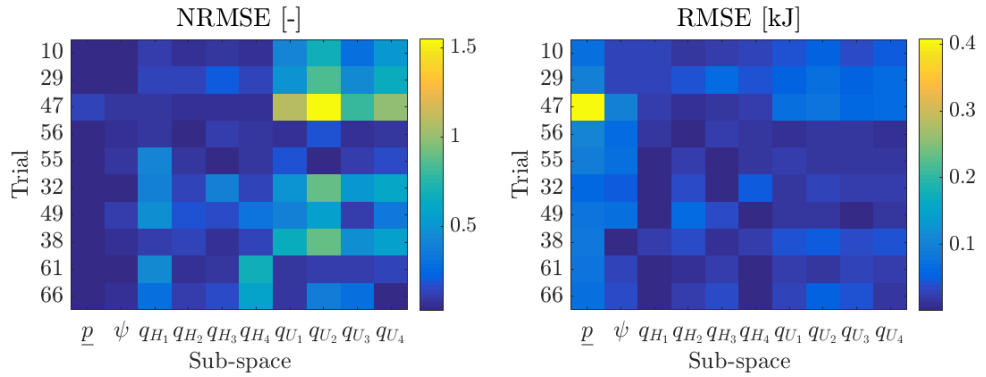
SunSpiral, V., Wheeler, D., Chavez-Clemente, D., and Mittman, D. (2012). Development and field testing of the FootFall planning system for the ATHLETE robots. *J. Field Robot.*, 22(3):483–505.

Vernaza, P. and Lee, D. D. (2012). Learning and exploiting low dimensional structure for efficient holonomic motion planning in high dimensional spaces. *Int. J. Robot. Res.*, 31(14):1739–1760.

Webb, D. J. and Van Den Berg, J. (2013). Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics. In *Proc. of IEEE ICRA*, pages 5054–5061.

Wermelinger, M., Diethelm, R., Kruesi, P., Siegwart, R., and Hutter, M. (2016). Navigation planning for legged robots in challenging terrain. In *Proc. of IEEE/RSJ IROS*, pages 1184–1189.

Wilcox, B. H., Litwin, T., Biesiadecki, J., Matthews, J., et al. (2007). ATHLETE: A cargo handling and manipulation robot for the Moon. *J. Field Robot.*, 24(5):421–434.



(a) The normalized root mean squared error (NRMSE) for each sub-space in each trial.

(b) The root mean squared error (RMSE) for each sub-space in each trial.

Figure 18: Comparison between the physical paths followed and the planned paths.

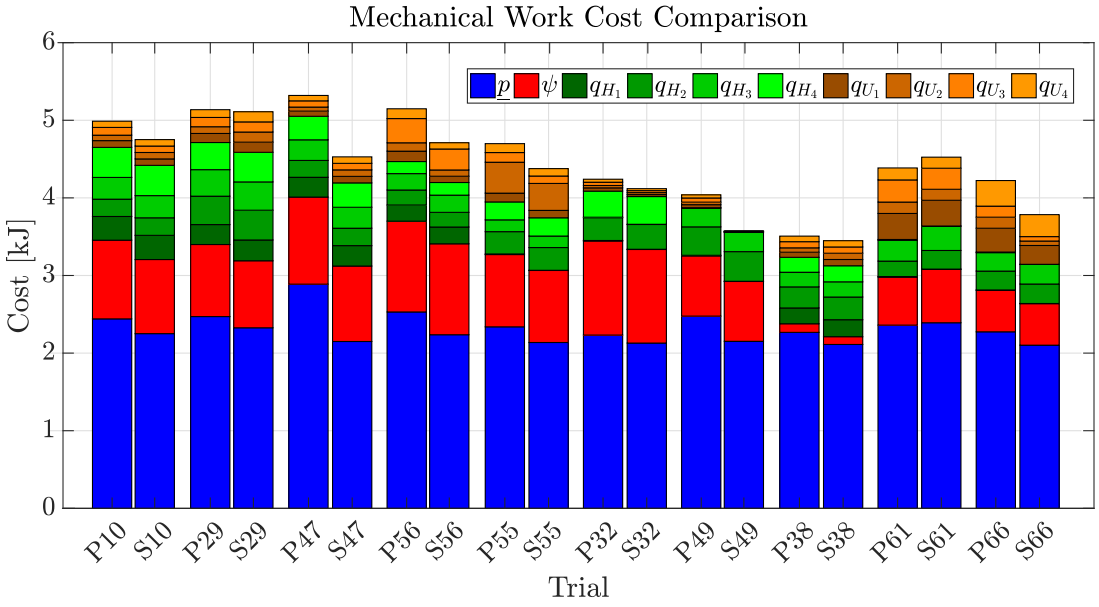


Figure 19: Successful traversal costs that are proportional to the translational and angular displacements of the MAMMOTH rover. Physical and simulated trials are compared side by side with 'P' denoting a physical trial and 'S' denoting a planned path.

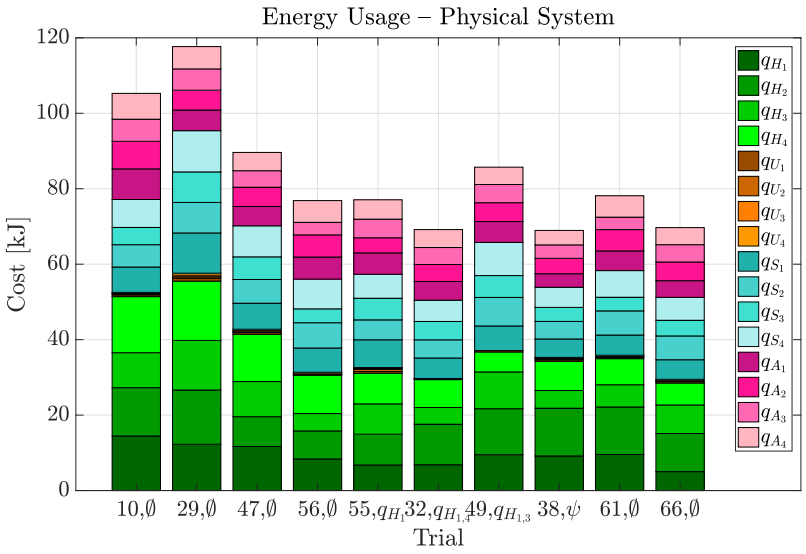


Figure 20: Energy cost calculated from current draw and voltage of each of the MAMMOTH rover's actuators.