Navigation on the Line: Traversability Analysis and Path Planning for Extreme-Terrain Rappelling Rovers

Michael Paton[1], Marlin P. Strub[2], Travis Brown[1], Rebecca J. Greene[1], Jacob Lizewski[1], Vandan Patel[1], Jonathan D. Gammell[2], and Issa A.D. Nesnas[1]

***Abstract*— Many areas of scientific interest in planetary exploration, such as lunar pits, icy-moon crevasses, and Martian craters, are inaccessible to current wheeled rovers. Rappelling rovers can safely traverse these steep surfaces, but require techniques to navigate their complex terrain. This dynamic navigation is inherently time-critical and communication constraints (e.g. delays and small communication windows) will require planetary systems to have some autonomy.**

Autonomous navigation for Martian rovers is well studied on moderately sloped and locally planar surfaces, but these methods do not readily transfer to tethered systems in non-planar 3D environments. Rappelling rovers in these situations have additional challenges, including terrain-tether interaction and its effects on rover stability, path planning and control.

This paper presents novel traversability analysis and path planning algorithms for rappelling rovers operating on steep terrains that account for terrain-tether interaction and the unique stability and reachability constraints of a rapelling system. The system is evaluated with a series of simulations and an analogue mission. In simulation, the planner was shown to reliably find safe paths down a 55 degree slope when a stable tether-terrain configuration exists and never recommended an unsafe path when one did not. In a planetary analogue mission, elements of the system were used to autonomously navigate Axel, a JPL rappelling rover, down a 30 degree slope with 95% autonomy by distance travelled over 46 meters.

I. Introduction

Areas of interest and high science return for *in-situ* robotic exploration of planetary bodies are often located in steep and rugged terrains. These include pits and caves that have the potential to harbor life or support human habitation, geological history in the exposed strata of vertical rock faces, climate history in steep icy slopes on Mars [1] and hypothesized similar locations on shadowed areas of the Moon and Mercury [2], and seasonal dark bands (Recurring Slope Lineae; RSL) on Martian slopes [3]. The ability to deliver instruments to these sites and perform long-duration measurements are key enabling technologies to further our understanding of the solar system.

These extreme terrains are either inaccessible or pose significant risk to current planetary rovers. First generation autonomy systems for Martian rovers (e.g., *Spirit*, *Opportunity*, and *Curiosity*) are designed for relatively flat terrain [4], [5] and have been used on slopes up to 30 degrees [6]. Next-generation systems (e.g., *Perseverance*) are designed for more challenging terrain but still have slope limits that preclude their use on extreme terrains that are highly sloped and/or non-planar.

Fig. 1: The JPL Axel rappelling rover traversing rocky vertical wall in a Martian analogue site in Mojave, CA. Rappelling rovers are capable of delivering large payloads to steep surfaces.

Potential robotic systems to access challenging terrain include flying [7], climbing [8], and rappelling [9] rovers. Rappelling rovers are attractive for scientific missions as their tethers allow for high payload-to-system mass ratios and can also extend operations into otherwise inaccessible areas when used to provide power and communications. Their operation in extreme terrain however often requires on-board autonomy.

Autonomy for rappelling rovers is challenging because of both the terrain and the platform. Highly sloped and/or non-planar terrains require complex stability analysis and path planning, often from onboard sensors of limited range. The tethered nature of rappelling rovers makes this analysis non-Markovian as the traversability and reachability of states depends on previous terrain-tether interactions, including intermediate anchor points.

This paper presents a system designed to consider these complexities and allow rappelling rovers to operate autonomously on rough and steep terrains. It includes tether-aware traversability analysis that predicts intermediate anchor points and considers them when evaluating stability. It also includes a sampling-based planner that uses this traversability analysis to plan paths on locally sensed 3D meshes.

This tethered navigation system was tested on JPL's Axel Rover [9], [10] in simulation and during an analogue mission. Axel is a simple and robust two-wheeled cylindrical rover

[1]Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Dr. Pasadena, CA michael.paton@jpl.nasa.gov

[2]Estimation, Search, and Planning (ESP) research group, Oxford Robotics Institute (ORI), University of Oxford, United Kingdom.

with an integrated tether that is controlled by an articulated boom. The simulation results demonstrate the system's ability to analyze terrain-tether interactions while searching for safe paths down a 55 degree slope. An early version of the system was also tested in a Martian analogue mission that demonstrated its capability to operate under realistic conditions (Fig. 1).

II. Related Work

Depending on the planetary body, science targets on steep terrains can be accessed using a range of approaches that include ascending, descending, and flying [11], [7]. On Mars, flying vehicles have limited payload capacity and short operational windows with which to make measurements. Climbing vehicles, such as Jet Propulsion Laboratory (JPL)'s Lemur robot [8], can access steep and even inverted terrain such as cave ceilings for long integration measurements. However, its gripping mechanism relies on surface properties for stability and requires complex articulated limbs, which reduce payload capacity. Rappelling rovers [9], [12], [13], [14], can also access steep terrain including vertical walls and cavernous pits using gravity but as such must begin at higher elevations. Rappelling rovers can use relatively simple designs, carry large payloads, and maintain power and communication with their base anchor, which could either be a lander or a rover with reliable communications to Earth [9]. Despite these mobility advantages, autonomous navigation for tethered rovers is complicated by tether-terrain interaction.

Path planning for tethered rovers have typically stayed within the $SE(2)$ state space and assumed that the environment is either planar or piecewise planar. ([15],[16]). Teshnizi et al. plans for the shortest path for a fixed-length tethered rover operating in $SE(2)$ with minimum bending radius and reachability constraints [17]. Zhang et al. coordinate motion for multiple tethered, planar robots operating in the same workspace. Operating in $SE(2)$, the planning algorithm plans for the shortest path while avoiding tether contact between rovers. Brass et al. plan paths in $SE(2)$ space while considering a rover with a finite tether length avoiding polygonal obstacles [18]. They further differentiate between rover's with self retracting tethers rovers that need to backtrack to pick up the tether. Tanner et al. plan for a tethered rover exploring the slope of a planetary crater. They represent the crater as a piecewise planar environment based on an orbital Digital Elevation Map (DEM) with 2D polygonal obstacles. The planner solves for the minimum distance $SE(2)$ path to the bottom of the crater and back that ensures the tether stays within a safe homotopy class. This homotopy class is the set of paths that ensure the tether does not wrap around obstacles and the rover is statically stable given the slope of the plane. While the latter work may be appropriate for generating global routes for terrains that can be approximated with piecewise planes, planning in $SE(3)$ is necessary for local route planning that have to account for centimeter-scale tether interactions with the environment as well as local slope changes. To achieve this, the rover must use a planner capable of operating in 3D non-planar space.

Traditional navigation methods that are aware of rover-scale hazards typically operate in $SE(2)$ state space or on 2.5D, pregenerated costmaps [4]. Path planning in $SE(3)$ is a common strategy for navigating Unmanned Aerial Vehicle (UAV)s [19]. However, these methods still largely rely on pregenerated obstacle maps. The non-Markovian nature of tether-based mobility makes precalculation of hazard maps computationally intractable.

For tethered navigation systems operating in $SE(3)$ in complex terrain, it is therefore required to perform "on demand" traversability assessment during planning. This allows the planner to consider state history and motion trajectories while planning. This methodology has been previously explored for untethered surface rovers. Howard et al. compute optimal trajectories for rovers by running full dynamics simulations on *a priori* global meshes [20]. Ono et al. approximate rocker-bogie limits based on "on-demand" wheel-surface contact analysis. Krussi et al. plan safe paths using sampling-based planning on unprocessed point-cloud maps [21]. It demonstrates computationally inexpensive planning in full 3D space. Motivated by this work, a similar methodology for planning paths in a full 3D space accounting for the complex constraints and terrain interactions of tethered robots was developed. To the knowledge of the authors, this is the first paper that presents 3D tethered planning.

III. Overview

The objective of the tether-aware planner is to find statically stable paths that minimize a weighted sum of yaw, roll, and path length. Such a path consists of a sequence of $SE(3)$ states that lie on the surface manifold of the terrain and are connected by in-place turns and straight-line drives.

States on the surface manifold are sampled by perturbing a randomly selected $\mathbb{R}^3$ grid point in the reconstructed 3D voxel space and combining it with a random rotation about the surface normal. This location and orientation is given to the rover settler which computes the associated $SE(3)$ state on the surface manifold.

The planner validates motions between states through on-demand traversability analysis that generates a series of interpolated states, and ensures that each state in the motion can be settled on the surface manifold, is statically stable, and does not result in collisions of the rover body with the terrain.

For each interpolated state, the traversability analysis computes the change in the sequence of points at which the tether interacts with the terrain (an *anchor history*) between itself and the previous state. This anchor history is used to determine whether the tether-force and the contact forces on of the rover's wheels can keep the rover statically stable. Because of this history-dependent traversability analysis, the $SE(3)$ states on the surface manifold are non-Markovian. This requires special care when sampling states and when rewiring the search tree. This tether-aware path planning is performed using novel changes toAdvanced Batch Informed

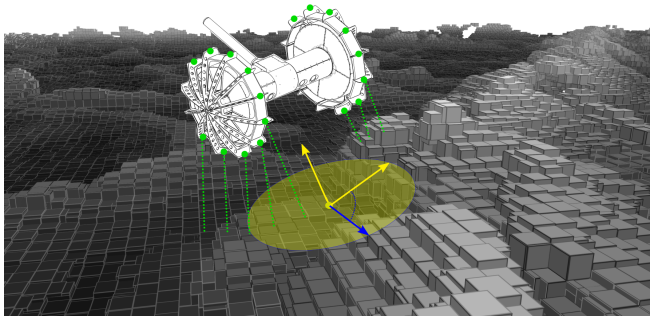


Fig. 2: Illustration of the Axel rover settling process. Given a point on the surface point cloud and a yaw angle, the settler seeks to find a full $SE(3)$ pose associated with the inputs. The settler uses the ESDF map to iteratively search for contact points on the wheels in the height and roll space of the rover.

Trees (ABIT*) to make it work in a non-Markovian setting (Section V).

IV. Traversability Analysis

The sampling-based planner uses the on-demand traversability analysis algorithms described in this section to determine the validity of randomly generated samples and the motion between them. To handle centimeter-scale rover- and tether-terrain interaction, the planner uses globally consistent Euclidean Signed Distance Field (ESDF) maps and surface point clouds generated by Voxblox [22]. For both cases, traversbility consists of the following logic checks: i) Did the rover properly settle on the terrain? ii) Are there any body collisions with the terrain? For motion validation, further tether-specific checks are made, given a motion between two states: i) Did the tether create any new anchors? ii) Did the tether detach from any old anchors? iii) Is the rover statically stable given the position of the tether anchor and the rover's contact points with the terrain? If these checks are passed then the sample is added to the map and motions are validated.

A. Rover Settling

Rover settling can be described as determining the $SE(3)$ transformation between the rover and the surface point cloud, T_{RM}, given a query point, $\mathsf{p}_m \in \mathbb{R}^3$, on the surface mesh, and a rotation, $_m \in SO(2)$, about the surface normal associated with p_m.

The settling process begins by calculating the surface normal, n, associated with the points surrounding p_m in an area equal to the footprint of the rover. Using the ESDF map, the direction of n is then set to point away from the interior of the surface. We initialize the unknown transformation, T_{RM}, by setting its origin to $\mathsf{p}_m + \mathsf{n}$, where is a user-provided parameter. This places the initial rover state above the surface. The algorithm then applies a rover-specific methodology to estimate the wheel contacts on the surface. The following is the methodology employed for the Axel rover.

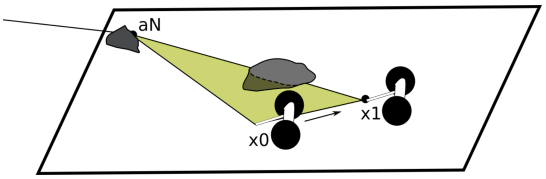


Fig. 3: Illustration of the Axel anchor-point prediction problem. This process involves solving for the attachment and/or detachment of tether-terrain contact points on the surface given an initial state X_0, and its anchor history $\mathsf{A_0}$.

Axel Specific Settling: An overview of settling for the Axel rover can be found in Fig. 2. The settling algorithm exploits the rover's rigid two-wheel body by iteratively searching for a distance along the surface normal and a body roll that places wheel contact points on the surface. Settling begins by approximating the rover's wheels contact points as a sparse point cloud. Each wheel is approximated by a circle of 12 contact points. At each iteration of the settling process, the contact points in the map coordinate frame are queried for their distances and gradients to the closest surface. The rover is dropped along the surface normal until one of the wheel contact points yields a distance within a threshold to the surface. Discrete perturbations to the the roll of the rover body are then searched until the two minimum contact points are roughly equal. The rover is again dropped onto the surface until the minimum contact points reach the surface. Once T_{RM} is determined, the body is checked for collisions. Collision points on the body are represented by a sparse point cloud on a line along the Y-axis of the rover. These collision points are transformed into the world frame by T_{RM} and queried in the ESDF map. If any of the points are within a distance threshold to the surface, then a collision is detected and the settling process fails.

The final step of the settling algorithms is to determine the $SO(2)$ rotation of the boom. Two boom angle configurations are considered: *inline* and *in-contact*. *Inline* corresponds to when the boom is inline with the tether as computed using the most recent anchor in the state's anchor history. *In contact* is when the boom is in contact with the ground, determined by sweeping through boom angles and checking for boom-terrain collisions. These boom configurations are used for both anchor detection and stability checks.

B. Anchor-Point Prediction

For rappelling rovers, the position of the most recent anchor point dictates the direction of tether force, a key factor in the ability of the rover to keep itself in force equilibrium. Thus, prediction of anchor point evolution is crucial for static stability analysis of poses within potential paths. The anchor-point prediction algorithm, illustrated in Fig. 3, seeks to compute the change in the anchor history for a given motion between two states.

The prediction algorithm is detailed in Algorithm 1. The input is an initial rover state, $\mathsf{X_0}$, with an associated anchor history $\mathsf{A}_0 = \{\mathsf{a}_0, \ldots, \mathsf{a}_N\}$, and an adjacent rover state $\mathsf{X_1}$. An anchor point, a_i, consists of two vectors in $\mathbb{R}^3$: the associated point in the world map, p, and its gradient, g,

which indicates the distance and direction to the surface. The output is the anchor history of X_1, A_1. To compute A_1 the following assumptions are made: i) the motion between X_0 and X_1 is relatively small, ii) the tether is taut at all times, iii) the tether between the last anchor point and the rover is a straight line (massless non-sagging tether), and iv) the anchor history is a stack of contact points; meaning anchor points must be removed in reverse sequential order. Field observations of tether behavior in relevant environments have indicated that these are reasonable assumptions and approximations. To properly estimate the change in anchor history between two states, the algorithm must track both anchor-point attachments (tether connects to the surface) and detachments (tether disconnects from the surface). The algorithm does not consider dynamic tether motions that would occur by applying a force to anchor points, such as tether slippage or destruction of terrain.

Algorithm 1: predict_anchors(X_0,A_0,X_1)

Result: The anchor history of X_1: A_1
$A_1 \leftarrow A_0$
$X_n \leftarrow X_0$
while continue **do**
  a_x = A_1.pop()
  a_{x-1} = A_1.back()
  if is_detached(a_{x-1}; a_x; X_n; X_1) **then**
    X_{n+1} = detach_line(X_n,X_1,a_x)
    A_{new} = attach_anchors(X_n,X_{n+1},a_x)
    if A_{new}.size() > 0 **then**
      A_1.push(a_x)
      A_1.append(A_{new})
    end if
  else
    A_{new} = attach_anchors(X_n,X_1,a_x)
    A_1.push(a_x)
    A_1.append(A_{new})
    break
  end if
end while

a) Anchor-Point Attachment: The attachment algorithm, illustrated in Fig. 3, searches for new anchor points generated by the motion between x_0 and x_1 in the ESDF map. We begin by defining the "sweeping space" of the tether. This is the 2D plane in the shape of a triangle (yellow shaded area) that the taught tether passes through when the rover translates from X_0 to X_1. To determine if there is any tether-terrain interaction while sweeping through this space, the algorithm uniformly samples the 2D plane to obtain interpolated distances and gradients to the surface from the ESDF map. These distances are then thresholded to obtain a list of obstacles in the world frame. The obstacles are then transformed to the coordinate frame of the 2D plane. Within the plane, the search for anchor points is a simple check on the angles between the current tether vector and the vectors between the obstacles and a_0. The obstacle with the smallest angle will be the surface point that the tether first comes into contact with. This point is added as a new anchor and the tether position is moved to be inline with the new point.

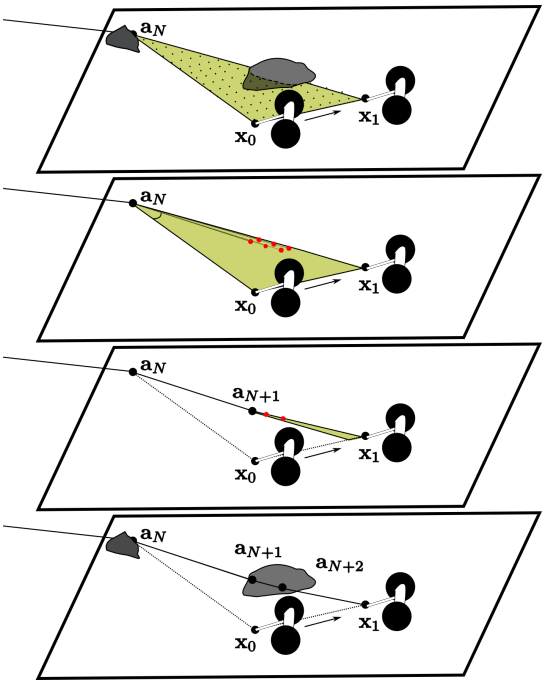


Fig. 4: Illustration of the Axel anchor point attachment algorithm.

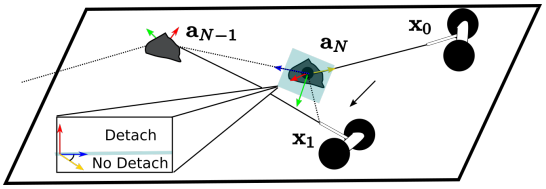


Fig. 5: Illustration of the Axel anchor point detachment algorithm.

The attachment process is then repeated with the new anchor point and current tether position until there are no obstacles found in the map. This results in a list of new anchor points with positions and gradients extracted from the map.

b) Anchor Point Detachment: The anchor point detachment algorithm is illustrated Fig. 5. Given an anchor history, $A = \{a_0 ::: a_N\}$, and the current rover state, X_1, the algorithm determines if the most recent anchor, a_N should be removed. This is determined by examining the position of the vector between X_1 and a_N in the coordinate frame, F_d, whose axes include the surface normal of the contact point of a_N (red arrow) and the vector between a_N and a_{N-1} (blue arrow). If the tether-anchor vector is above the XY plane of this frame, it is considered a detachment event. In the illustration, the state X_0 does not contain a detachment while the state X_1 does.

C. Tether-based Stability

Once the in-contact pose, N points of surface contact, and the state's anchor history have been determined, the pose stability is evaluated using constrained quadratic programming. This consists of searching for feasible contact and tether forces that place Axel in static equilibrium (Fig. 6). While the settler treats ground contact as a point intersection, the actual contact generally consists of a finite-sized contact patch enclosed by a pair of grousers. This allows each contact

"point" to carry moments between Axel and the ground. This property is fundamental to Axel's ability to perform certain maneuvers such as lifting the boom off of the ground. Thus, each point i is allowed to apply a full wrench $(\boldsymbol{\lambda}_i; \boldsymbol{M}_i)$ subject to

$$\boldsymbol{\lambda}_{i;z} \quad 0 \tag{1}$$

$$j\boldsymbol{\lambda}_{i;x}j \quad \boldsymbol{\lambda}_{i;z} \quad 0 \tag{2}$$

$$j\boldsymbol{\lambda}_{i;y}j \quad \boldsymbol{\lambda}_{i;z} \quad 0 \tag{3}$$

$$j\boldsymbol{M}_{i;x}j \quad {}_m\boldsymbol{\lambda}_{i;z} \quad 0 \tag{4}$$

$$j\boldsymbol{M}_{i;y}j \quad {}_m\boldsymbol{\lambda}_{i;z} \quad 0 \tag{5}$$

$$j\boldsymbol{M}_{i;z}j \quad {}_m\boldsymbol{\lambda}_{i;z} \quad 0 \tag{6}$$

where contact frames are defined with the z-axis aligned along the contact normal and $\boldsymbol{\lambda}_i$ and $\boldsymbol{M}_i$ represent the generalized forces and moments at the i^{th} contact point. Friction cone constraints are approximated with friction pyramids, allowing them to be treated as a set of linear inequalities. Moment constraints follow the same form, with bounds defined as the normal force times a scalar ${}_m$. Static equilibrium is also treated as a constraint, with force and moment balance given by

$$\sum \boldsymbol{F} = m_1\boldsymbol{g} + m_2\boldsymbol{g} + R_t\boldsymbol{\lambda}_t + \sum_{i=1}^{N} R_{ci}\boldsymbol{\lambda}_i = 0 \tag{7}$$

$$\sum \boldsymbol{M} = (\boldsymbol{r}_1 \quad m_1\boldsymbol{g}) + (\boldsymbol{r}_2 \quad m_2\boldsymbol{g}) + (\boldsymbol{r}_t \quad R_t\boldsymbol{\lambda}_t) + \sum_{i=1}^{N} \boldsymbol{r}_{ci} \quad R_{ci}\boldsymbol{\lambda}_i = 0 \tag{8}$$

where $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ locate the center-of-mass of the two bodies, $\boldsymbol{r}_t$ locates the tether exit orifice, and $\boldsymbol{r_{ci}}$ locates the ith contact point. While simultaneous solution of these constraints would indicate stability, it wouldn't show how sensitive the stability is to the friction parameters, which have high uncertainty. Thus, a quadratic objective function is defined which quantifies friction and moment cone:

$$f_{obj} = \sum_{i=1}^{N} \frac{\boldsymbol{\lambda}_{i;x}^2 + \boldsymbol{\lambda}_{i;y}^2}{2} \quad 2\boldsymbol{\lambda}_{i;z}^2 + \sum_{i=1}^{N} \frac{\boldsymbol{M}_{i;x} + \boldsymbol{M}_{i;y} + \boldsymbol{M}_{i;z}}{\frac{2}{m}} \quad 3\boldsymbol{\lambda}_{i;z}^2 \tag{9}$$

This drives the solution to the set of forces/moments with maximum margin on friction and moment cone utilization. Cone utilization forms a natural stability metric with which poses can be compared. Because the constraints are linear and the objective function is quadratic, stability can be determined very efficiently using off-the-shelf QP solvers. In this work, a variety of solvers were tested, with NLOPT [23] generally providing the best performance.

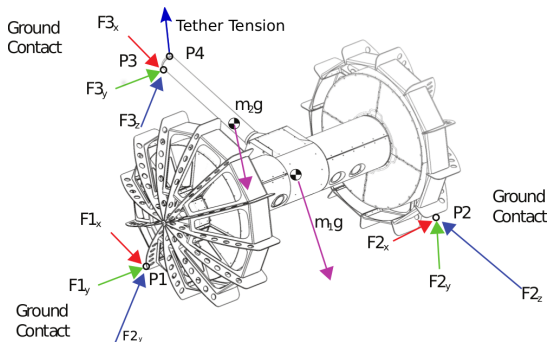


Fig. 6: Force body diagram for the Axel Rover. Tether-dependent static stability of the system depends on terrain contact points with the wheels (P1,P2), potential contact with the boom and the ground (P3), and the tether forces on the end of the boom (F3), and the tether tension applied. Correct static stability requires a knowledge of the position of the last anchor point.

V. Tether-aware Path Planning

The rover settling, anchor prediction, and the tether-aware stability analysis are used to determine whether states and motions between states are valid. This information can be used by sampling-based planning algorithms to find valid paths between start and goal states.

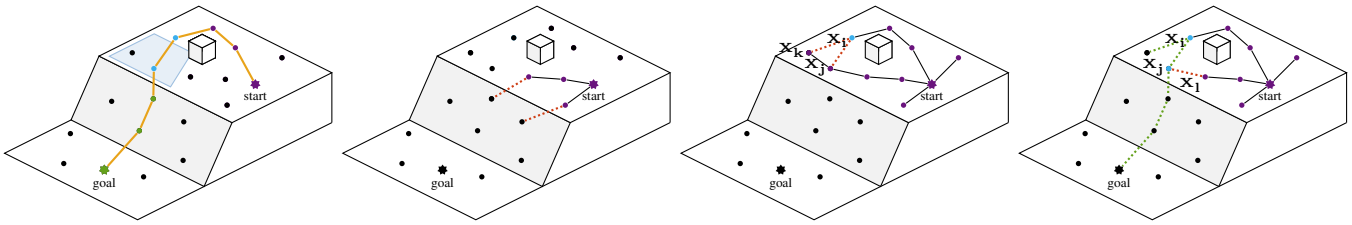
We used an Open Motion Planning Library (OMPL) [24] implementation of ABIT* [25], an almost-surely asymptotically optimal sampling-based planner that can avoid the expensive evaluation of unnecessary edges and reduces initial solution times compared to Batch Informed Trees (BIT*) [26]. It views the planning problem as the two subproblems of approximation and search, which allows it to process an anytime sampling-based approximation using advanced graph-search techniques.

ABIT* builds an approximation of the state space by sampling multiple batches of states and viewing these states as an increasingly dense edge-implicit Random Geometric Graph (RGG) [27]. Once an initial solution is found, ABIT* only increases the density of its RGG approximation in the region of the state space that can possibly improve the current solution by using informed sampling [28].

The implicit edges of the RGG approximation are processed in order of their (inflated) potential solution quality. The potential solution quality of an edge is taken as the sum of the current cost-to-come from the start to the edge's parent state, an estimate of the edge cost, and an estimate of the cost-to-go from the edge's child state to the goal.

ABIT* prioritizes quickly finding an initial suboptimal solution to the current approximation over efficiently finding the resolution optimum. It does this by inflating the cost-to-go estimates and then decreasing the inflation to tighten the optimality bound, as in anytime graph-search algorithms. It does this efficiently by tracking changes to state connections, as in anytime repairing search algorithms. The search of each sampling-based approximation is stopped based on a suboptimality bound, as in truncated graph-search algorithms, to avoid the computational effort of fully searching an approximation that will change.

These advanced graph-search techniques allow ABIT* to



(a) (b) (c) (d)

Fig. 7: An illustration of a planning problem that must consider the non-Markovian nature of tether constraints to plan for a new anchor and find a solution (a). Axel is instructed to move from the start down the slope to the goal but the initial anchor history does not support a direct descent (b) and solutions must first anchor on the column. As states in the blue region can be reached with lower cost by passing in front of the obstacle then a naive planner will connect all potential solution states, $\mathbf{x_j}$ and $\mathbf{x_k}$, with an anchor history that does not support a solution. Once connected, these states cannot be rewired to states that support a solution but have incompatible anchor histories, $\mathbf{x_i}$ (c). Sampling states with both undefined and defined anchor histories ensures that multiple anchor histories can expand into a region and allows for ABIT* to find safe path down the steep slope (d).

balance the exploitation of its approximation (i.e., repairing the search) with the exploration of the state space (i.e., increasing the density of the approximation). This reduces the number of edges evaluated and finds initial solutions quickly, which makes ABIT* particularly well-suited to the planning problems posed by Axel, where edge evaluations are computationally expensive due to the traversability analysis. The full details of ABIT* are in [25].

A. ABIT* with non-Markovian States

Axel requires paths as sequences of $SE(3)$ states settled on the surface manifold of the terrain. The stability of these states and the feasibility of motion between them depends on their anchor histories, making this a non-Markovian planning problem. This non-Markovian traversability analysis complicates sampling-based planning in particular, tree rewiring.

Anchor histories must be considered both when sampling and connecting states. By default, the anchor history of newly sampled states is defined when they are first connected to the tree using the anchor prediction algorithm described in Section IV-B. This ensures that new states are compatible with the existing tree but may bias the tree with an anchor history that does not support a solution, as in Fig. 7c. To avoid this, a user-defined percentage of new states are sampled with a predetermined anchor history copied from their neighbours. This maintains the exploration of all known anchor histories and allows ABIT* to plan different paths with multiple incompatible anchor histories in one region and therefore find solutions that require specific anchor histories, as in Fig. 7d.

Rewirings of states to parents with incompatible anchor histories must be prevented because each state's stability depends on all of its ancestors' anchors. If the anchor history of one state changes, then the stability analysis of all of its descendent states and motions would have to be recalculated, potentially invalidating previously valid states. Whenever ABIT* finds a better path to the goal, a new goal state is created with undetermined anchor history. This ensures that there is always a goal state that the next better path can connect to.

VI. Experimental Setup

A. Simulation Tests

This section describes the simulations to validate the core concepts of the tether-based path planner. An overview of the three tests conducted in this experiment can be found in Fig. 8. The maps used for these tests were generated using the Voxblox simulation environment [22] and consists of flat ground, a 10 5 4 m ledge with a 55 degree slope on one side. In test 3, the map is slightly altered to add a tall pole in the middle of the ledge. For each test, we run 100 instances of a fixed planning problem where the start position is on the ledge facing towards the slope and the goal is on the flat ground. Between tests, the environment and initial anchor positions are varied to demonstrate the capabilities of the planner.

In test 1, we initialize the root anchor to be 4.9 m behind and 2.0 m above the rover allowing the area of the slope directly in front of the rover to be stable. This is indicated as a green shaded area of the slope in the left of Fig. 8. The outcome of this test is expected to yield paths that are direct connections between the start and the goal.

In test 2, we replicate the conditions of test 1, but move the anchor so it is 8.2 m in front of the rover and 2.0 m above the rover. This configurations makes the entire 55 degree slope unstable. This can be seen in the middle of Fig. 8 as a red shaded area of the slope. The outcome of this test is expected to yield zero paths found from the planner.

In test 3, we replicate the conditions of test 2, but add a tall pole to the middle of the map. This configuration makes one side of the slope stable if and only if the path loops around the back of the pole to add an intermediate anchor point before descending the slope. This can be seen in the right of Fig. 8 as a yellow shaded area of the slope. The outcome of this test is expected to yield paths that add this intermediate anchor before descending.

B. Martian Analogue Experiment

An early version of the system that does not include tether constraints was validated during a week-long field test at a Martian analogue site at the Golden Queen Mine in Mojave,

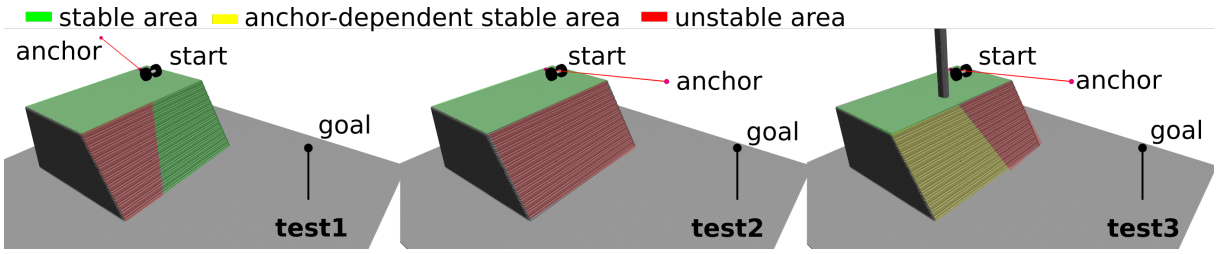


Fig. 8: Overview of the planner simulation experiments. Our testing environment consists of a 4 m ledge with a 55 degree slope. Each tests maintains the same start and goal position while varying the anchor start location and the environment. Green shaded areas represent locations where the initial anchor allows stable traversal. Yellow shaded areas represent locations where an intermediate anchor will allow stable traversal. Red shaded areas represent locations that are unstable for the initial anchor location.

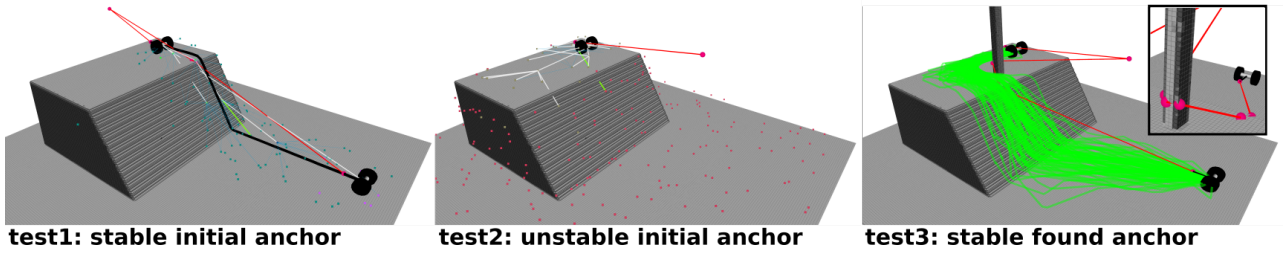


Fig. 9: Simulation results for planning with tether constraints. In test 1 we demonstrated the planner's ability to plan safe paths when there is a stable anchor. In test 2 we moved the anchor to an unstable position and demonstrated the planner's ability to recognize the slope as hazardous. In test 3 we added a pole and demonstrated the planner's ability to find intermediate, stable anchor points.

CA. The experiment demonstrated the planner's ability to operate on noisy, uncertain and short-range maps generated from an onboard stereo camera, and plan paths that avoid body collisions and slope thresholds. During these tests, the planner operates on short-range maps by allowing for goals to be outside of the boundaries of the map and connecting states to this off-map goal if they are close to the edge of the map. During this field test we generated local voxblox maps using an onboard stereo vision pipeline.

VII. Results

A. Simulation Tests

Summary results of the simulation tests can be found in Table I. For each test we ran 100 trials of the same planning problem while varying the random seed. For each problem we ran the planner for 750 iterations. For test one, the planner found the straight line path between the start and the goal for all 100 trials in the first iteration. For test two, we moved the anchor to an unstable location. For each test, after 750 iterations the planner was unable to find a safe path to the goal. These two tests in conjunction with each other demonstrate the importance of anchor point placement on traversability. In the final test we add a pole to the middle of the map. In this configuration we expect the planner to find a path that wraps around and anchors to the pole before descending. Out of the 100 trials performed, 84 of the tests found paths in the stable homotopy class of anchoring to the pole and 16 of the tests resulted in no paths found.

Paths for each test can be seen in Fig. 9. These figures show the start and goal positions of the robot, and their respective anchor histories. Green lines indicate found paths. For test1, it shows that the planner recommended the straight line down the slope and an intermediate anchor was found on the lip of the slope as the rover descended. For test2, it shows that ABIT* has correctly failed to find a safe path to the bottom. For test3, it shows that the path has the rover drive around the pole, generating an intermediate anchor point that allows it to drive directly down the slope.

TABLE I: Simulation Results

	Found Paths (/100)		Initial Solution (;)	
	Stable	Unstable	Time [s]	Iterations
Test 1	100%	0%	(0:29; 0:05)	(1; 0)
Test 2	0%	0%	—	—
Test 3	84%	0%	(12:8; 10:3)	(188; 141)

Through these tests we demonstrated that our planner is able to generate theoretically safe paths in while considering while considering the unique safety constraints imposed on a tethered rappelling platform.

B. Martian Analogue Experiment

Autonomy results for the Martian analogue experiment are illustrated in Fig. 10. Over the course of the 46 m traverse, the rover drove 44 m autonomously (white path segments) with intermittent manual interventions (black segments). These experiments do not test the complete system but show that i) terrain assessment can operate on noisy data collected from an onboard stereo camera, ii) the rover settling calculations, body-collision checks, and slope threshold checks are consistent with reality, and iii) the planner can successfully navigate towards a long-distance goal. Manual interventions were primarily the result of failures in the path-tracking controller to account for tether forces which required the human operator to intervene to drive the planned path. Failures in the settler as a result of sensor noise were also observed at the beginning of the traverse.