2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

Online Motion Planning for Tethered Robots in Extreme Terrain

Melissa M. Tanner[1], Joel W. Burdick [1], and Issa A. D. Nesnas[2]

Abstract— **Several potentially important science targets have been observed in *extreme terrains* (steep or vertical slopes, possibly covered in loose soil or granular media) on other planets. Robots which can access these extreme terrains will likely use tethers to provide climbing and stabilizing force. To prevent tether entanglement during descent and subsequent ascent through such terrain, a motion planning procedure is needed. Abad-Manterola, Nesnas, and Burdick [1] previously presented such a motion planner for the case in which the geometry of the terrain is known a priori with high precision. Their algorithm finds ascent/descent paths of fixed homotopy, which minimizes the likelihood of tether entanglement. This paper presents an extension of the algorithm to the case where the terrain is poorly known prior to the start of the descent. In particular, we develop new results for how the discovery of previously unknown obstacles modifies the homotopy classes underlying the motion planning problem. We also present a planning algorithm which takes the modified homotopy into account. An example illustrates the methodology.**

I. Introduction

The recent successful landing of the Mars Science Laboratory (MSL) Curiosity rover reminds us of the potential for robots to explore extraplanetary terrains. To date, NASA's Martian rovers (Sojourner, Spirit, Opportunity, and now Curiosity) have used a common mobility platform–the six-wheeled rocker-bogie suspension design [2]. While it has been estimated that this type of platform can operate on $\sim 60\%$ of the Martian surface, some of the most interesting science targets occur in currently inaccessible terrains. For example, in 2011, seasonal dark streaks were discovered on the slopes of Mars' Newton Crater (see Fig. 1). These *recurring slope lineae* (RSL) are currently hypothesized to be flows of briny water caused by the warmer Martian summer. An extreme terrain rover could access these slopes to examine and sample the RSL at close range, potentially confirming the presence of a life-supporting aqueous environment on Mars. Beyond Mars, cold traps found at the bottom of deep craters on the Moon might harbor water, crucial to a future manned base. Nesnas et al. [3] provides a list of other representative extreme extraplanetary terrains and their associated scientific targets.

In light of this potential need for extreme terrain rovers, the Jet Propulsion Laboratory (JPL) and the California Institute of Technology collaborated to design and develop the *Axel* class rover, and its *DuAxel* rover configuration. Axel is a two-wheeled robot that can rappel down steep terrain using a tether in order to carry out science at close range (see Fig. 2). The tether is routed through the arm, and wound around a reel in the center of its body. A solo Axel is conceived as a daughter ship in a mother-daughter configuration, with Axel's tether anchored in the mother ship (a rover or lander at the top of the extreme terrain) [3]. In addition to providing mechanical support, the tether can provide power and communication. Because the Axel body acts as a winch, paying out or reeling in the tether as it travels, the system minimizes tether abrasion.

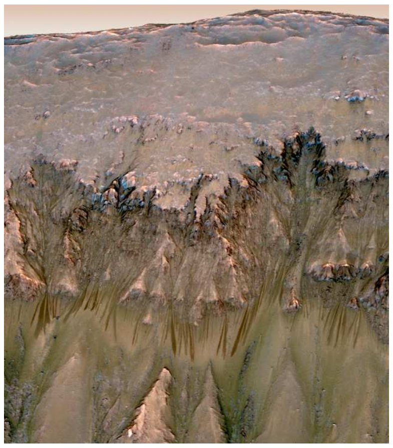
Fig. 1: Orbital imagery of dark briny flows on the Martian surface. The ability to measure and *in-situ* sample these flows on their native steep slopes could confirm the presence of life-supporting water on Mars [4].

Axel is minimally actuated, having a motor for each wheel, one to spool the tether, and one to control the arm through which the tether passes. This combination of motors allows Axel to rotate its body independently of its wheels, enabling it to take pictures or deploy scientific instruments from the access panels in its instrument modules while on a steep cliff face. In the DuAxel configuration, two Axel-class rovers dock with a *central module*, forming a 4-wheeled rover which can traverse long distances to the edge of an extreme terrain area, whereupon one or both Axels disengage from the central module to descend into the hazardous area (see Fig. 3). The central module acts as an anchor and mother ship. For more details on the geometry and capabilities of the Axel and DuAxel rovers, see Nesnas et al. [3]. Axel is not the first tethered robot to be developed for steep terrain access. Nesnas et al. [3] also compares Axel to other extreme terrain robots, such as Dante, Athlete, Cliffbot, and Lemur.

Figure 4 shows Axel descending steep terrain during a

*This work was supported by the Keck Institute for Space Studies. In addition, the third author was supported by the JPL RTD program.

[1]M. Tanner and J. Burdick are with the Department of Mechanical and Civil Engineering, California Institute of Technology, Pasadena, CA 91125 USA mmtanner@caltech.edu and jwb@robotics.caltech.edu

[2]I. Nesnas is with the Jet Propulsion Laboratory, Pasadena, CA 91109 USA nesnas@jpl.nasa.gov

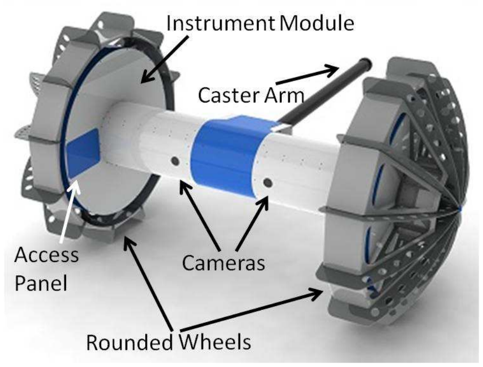

Fig. 2: CAD rendering of the Axel rover.

Fig. 3: The DuAxel rover on rough terrain. DuAxel consists of two Axel-class rovers docked with a central module.

field test in the Arizona desert. Clearly, without thoughtful planning of Axel's descent in such terrain, there is a high propensity for its tether to become entangled on terrain features. Most previous efforts to develop motion planning algorithms for tethered robots exclusively considered the tether a mere umbilical providing power, communication, and control signals. In these studies, the tether did not generate the large reaction forces needed for mobility. The primary goal of these algorithms was to minimize entanglement of the trailing tether with obstacles or other robots. For example, Hert and Lumelsky considered path planning with multiple tethered robots [5]–[8]. Xavier found the shortest path for a tethered robot [9]. Axel, however, must use its tether for support and to generate climbing forces or stabilizing forces, not just for power and communication. The forces provided by the tether must be accounted for in the motion planning process, and the interaction of the tether with the terrain under load must also be considered.

To date, the only algorithm for planning the motion of robots whose tethers generate mobility and safety forces has been presented by Abad-Manterola et al. [1]. This method, whose underpinnings are reviewed in Section II, is based on two key observations about tether-based mobility in extreme terrain:

1) A *necessary* condition to avoid entanglement of the tether during a round trip descent/ascent cycle is to ensure that the descent and ascent paths lie in the same path homotopy class. When this constraint is violated, the tether can become looped around an obstacle.
2) In the course of vehicle ascent, the robot will pull the tether taut, forming the *shortest homotopic path* (SHP). This taut tether configuration must be homotopic to both the ascent and descent paths, and therefore can be used as a canonical representation of a homotopy class.

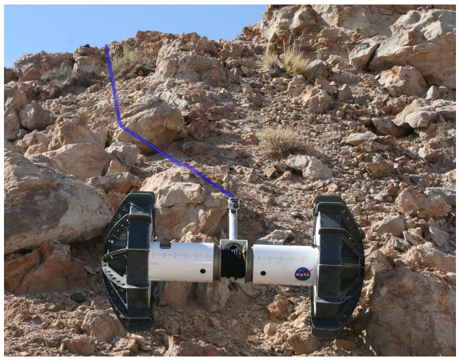

Fig. 4: A picture of Axel descending steep, rough terrain in the Arizona desert. Tether highlighted in blue.

More importantly, the planning method presented in Abad-Manterola et al. [1] assumes the geometry of the terrain and its obstacles are known a priori. While it is true that high resolution orbital imagery will almost certainly be available prior to the descent of an extreme terrain vehicle, the resolution of this imagery cannot be sufficient to identify all obstacles to the robot's progress. It must be assumed that unforeseen obstacles will arise during the robot's traverse, and also that orbital imagery will contain errors or misalignments.

This paper extends the method from Abad-Manterola et al. [1] to develop an algorithm which can plan tethered robot motions in an extreme terrain whose geometry is not fully known. The key to this approach is the observation that newly observed obstacles and terrain features found during descent have limited, and well-definable, effects on the special triangulation which is the basis for managing the homotopy classes of the ascent/descent paths.

Section II reviews the algorithm developed in Abad-Manterola et al. [1] in order to set the stage for our new contributions. Section III shows how newly discovered obstacles or terrain features found during the descent stage affect the originally constructed plan, and proposes an online algorithm to accommodate new terrain features. Section IV presents an example using this algorithm. Section V concludes with a summary of open issues.

II. Background: Tethered robot path planning in known terrains with homotopy constraints

The permanently-shadowed depths of the lunar Shackleton Crater form a cold trap, hypothesized to contain volatile comet remnants such as water ice. Figure 3 in Abad-Manterola et al. [1] shows a side view elevation map of Shackleton Crater's walls, based on altimetry from NASA's Lunar Reconnaissance Orbiter. Like many extraplanetary science targets, the rim and bottom of the crater are relatively

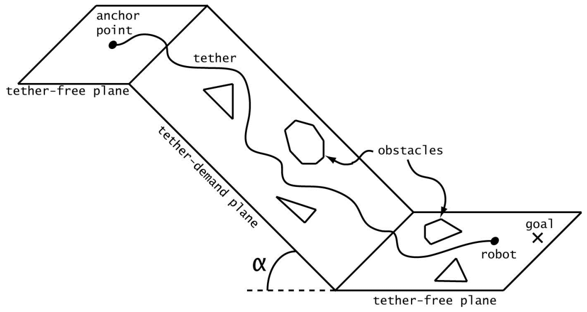

Fig. 5: Simplified model of an extreme terrain: A tether-demand plane interposed between two tether-free planes [1].

flat, while the crater walls are composed of stretches of consistent steeper slopes. We can thus idealize such terrains as a series of planes strewn with obstacles, which we approximate as polygons (see Fig. 5). The steep slopes are termed *tether-demand planes*, as the tether must generate climbing forces when the robot operates on these steep slopes, while the gentle slopes are termed *tether-free* regions (where the rover can traverse without the need for a tether).

Experiments with Axel and DuAxel in rough terrain have shown that tethered *ascent* over steep and rocky slopes is generally more difficult than tethered descent over the same terrain [3]. Working against gravity, steep terrains covered by loose soil or granular material often afford little traction, and thus an ascending tethered robot can easily become stuck underneath an obstacle that was easily traversed during the descent. Since not all feasible descent paths will have a corresponding feasible ascent, the planning procedure of Abad-Manterola et al. [1] is based on a two step approach: find safe ascent paths, and then search for feasible descent paths that are homotopic to those safe ascent paths.

Let $\mathbf{a_0}$ denote the location of the *anchor point* where the distal end of the tether is anchored, enabling Axel to descend into steep terrain. Let $\mathbf{g}$ denote the location of the goal.

Consider the robot's ascent on a tether-demand plane. The robot reels in the tether until it is taut, thereafter using the cable tension to create ascending forces. If we assume that the terrain between the obstacles is frictionless, then the tether will necessarily take the shortest obstacle-free path between the robot and the anchor point, potentially contacting the boundary of one or more obstacles. Hence, one can compute the configuration of a taut tether, given its slack configuration, by finding the *shortest homotopic path* (SHP) from the robot's configuration to $\mathbf{a_0}$.

There will generally be more than one way to thread a taut tether around obstacles between the anchor and the goal. Each of the possible windings of the tether through the obstacles represents a different path *homotopy class*. Hence, the path planning process starts with identifying all of the possible path homotopy classes between the anchor and the robot's goal.

A. Boundary Triangulated Manifolds

To find all of the shortest homotopic paths (corresponding to possible tether routes between $\mathbf{a_0}$ and $\mathbf{g}$ that do not wind around an obstacle), one approach first constructs a *Boundary Triangulated Manifold* (BTM) of the tether-demand planes.

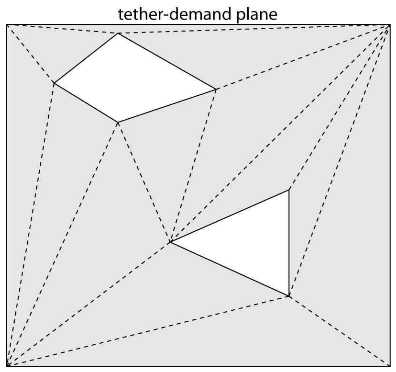

Fig. 6: Example of a Boundary Triangulated Manifold (BTM) [1]. Obstacles shown in white.

Definition 1: Let P be a polygonally bounded region of $\mathbb{R}^2$, which may contain one or more polygonal holes (obstacles), $\{O_i\} = O_1, \ldots, O_N$. A **Boundary Triangulation** of $P \backslash \{O_i\}$ consists a decomposition of $P \backslash \{O_i\}$ into *boundary triangles.* The vertices of a boundary triangle must be incident to only two *boundary edges*. Boundary edges are incident to only a single triangle. A **Boundary Triangulated 2-Manifold** (BTM) is a 2-dimensional simplicial complex in which all vertices are boundary vertices.

Practically speaking, boundary edges form the boundaries of the tether demand regions, or the edges of obstacles. The vertices of the triangles in a BTM must always lie on vertices at the end of boundary edge segments. De Berg [10] gives algorithms to break a two-dimensional map into y-monotone pieces, and to triangulate those pieces to form a BTM. Note that there is not a unique boundary triangulation of a polygonal region with obstacles. Figure 6 shows an example of a BTM.

Sleeves. Each homotopy class can be represented by a *sleeve,* a polygon formed by those triangles through which a SHP passes (see Fig. 7) [11]. Any path entirely within the sleeve is homotopic to any other path within it, although the set of homotopic paths is not limited to those in the sleeve. In addition, paths that travel outside of the sleeve and then return through the same exiting edge without looping around any obstacles are also homotopic to every path within the sleeve.

We use Hershberger and Snoeyink's algorithm to find the SHP in a sleeve of a BTM [11]. This is based on earlier work by Chazelle [12] and Lee and Preparata [13], who wrote funnel algorithms to compute the shortest path in a polygon. Recently Bespamyatnikh [14] introduced an algorithm that can compute the SHP of a simple path in reduced time. Abad-Manterola et al. [1] present a method to find all unique sleeves of a BTM that contain the anchor and goal points.

Anchor points. *Intermediate anchor points* are the points at which the taut tether (the SHP of the given sleeve) contacts one or more of the obstacles, $O_1, \ldots, O_N$. The intermediate anchor points $a_1, \ldots, a_k$ are indexed in increasing order along the tether from that anchor point. An intermediate

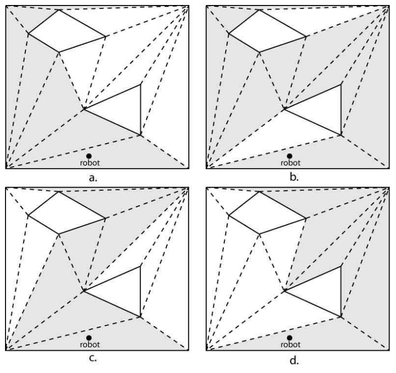

Fig. 7: Example of *sleeves*, in gray, in a Boundary Triangulated Manifold [1].

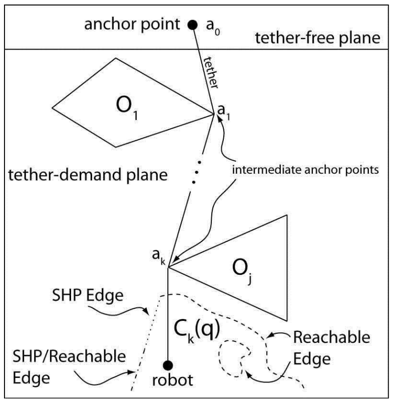

Fig. 8: Example of anchor points and SHP edges. The dotted line surrounds the controllable set [1].

anchor point, $\mathbf{a_j}$, is *passable* from robot configuration q if, given q and an SHP with anchor points $a_1, \ldots, a_j$, the robot can reach a configuration which removes $\mathbf{a_j}$ from the SHP and makes $\mathbf{a_{j-1}}$ the proximal anchor point. In other words, if the robot can maneuver so as to remove its tether from contact with the obstacle at that anchor point, that anchor point is considered passable. See Abad-Manterola et al. [1] for a procedure to assess passability.

One example is illustrated in Figure 8. For some intermediate anchor point $\mathbf{a_k}$, $C_k(q)$ represents the reachable set, the points that are reachable from the robot's current configuration, given the wheel-terrain interaction model. As it crosses the SHP edge, the robot removes $\mathbf{a_k}$ from the list of intermediate anchor points, leaving $\mathbf{a_{k-1}}$ as the proximal intermediate anchor point. Therefore, $\mathbf{a_k}$ is passable. The robot must then compute $C_{k-1}(q)$ to determine if the next intermediate anchor point is passable, continuing up the line. If all of the anchor points are passable, then a feasible ascent path exists.

B. Using Complex Analysis to Find Multiple Homotopy Classes

L-values. As a requirement for online path planning, we add a homotopy feature not found in Abad-Manterola et al. [1]. Bhattacharya et al. [15] describe a method, based on the Cauchy Integral Theorem from complex analysis, to check if two paths are homotopic. While there are alternate methods to test for homotopy, such as that in Cabello et al. [16], this complex analysis method also lends itself well to traditional graph search. Bhattacharya et al. model (x, y) nodes on a directed graph in the complex plane, as $z = x+iy$. Each obstacle $O_1, ..., O_N$ is represented by a pole placed somewhere inside that obstacle, ζ_i. Since (by Cauchy's Integral Theorem) the integral of an analytic function around a simply connected region is 0, and around a pole is some non-zero multiple of $2\pi i$, we can perform a contour integral to determine if there is an obstacle inside the loop formed by two paths. If there is no obstacle, the contour integral is zero and the two paths are homotopic.

Therefore, we can compare the contour integral for one path with that of another path to determine if the two paths are homotopic. The contour integral for each path is called its *L-value*, given below:

$$L(\tau) = \int_{\tau} \frac{f_0(z)}{(z-\zeta_1)(z-\zeta_2)...(z-\zeta_N)} \mathrm{d}z$$

where τ is the path, and $f_0(z)$ is any analytic function over the complex plane. If two paths (oriented in the same direction, from $\mathbf{a_0}$ to $\mathbf{g}$) have the same L-value, they are homotopic. Therefore, Bhattacharya et al. [15] show that one can check the homotopy of two paths using a contour integral over any analytic function. They use homotopy as a constraint in search, allowing one to limit search to a certain homotopy class, or rule out other homotopy classes.

We can use this L-augmented search method to quickly find a number of sleeves in a given BTM. We start by choosing some heuristic, such as distance. Then we search for the least-costly route using A*, which uses an open list for visible paths. Although any informed search algorithm or evolutionary algorithm can be used, A* was chosen for its simplicity and bounded time structure. Having found one path to the goal, we continue the search, constraining new nodes to have an L-value different from that of the found path. By continuing in this manner, we can build up a list of paths, each in a different homotopy class. In addition, by making each of the graph nodes a triangle in the BTM, we both simplify our search and make it very easy to find the associated sleeve, given a path from $\mathbf{a_0}$ to $\mathbf{g}$.

C. The Pre-Planning Algorithm

Our path planning algorithm starts similarly to that given by Abad-Manterola et al. [1]. We first triangulate the tether-demand plane(s) into a BTM. Without considering kinodynamics, we then apply an L-value constrained A* search until we find a path to the goal. We use this path to identify a sleeve, as classified by its L-value. Using Hershberger and Snoeyink's algorithm, we find the SHP for the sleeve, and identify the anchor points. After checking if the desired sleeve has a feasible path, we choose ascent and descent paths from the set of feasible paths. We can ensure backtrackability if we choose a descent path from the set of feasible ascent paths, such that the robot can always return to its anchor point at the top. If there is no feasible path in this sleeve, we add this sleeve's L-value to the list of constraints and continue the A* search in regions of different L-values. This continues until we find a feasible path, or terminate without finding any path.

Algorithm 1 Offline Pre-Planning Algorithm

```
    find BTM
    $L_c \leftarrow \emptyset$ {Initialize list of L-value constraints}
    $updates \leftarrow \emptyset$
    $feasiblePaths \leftarrow \emptyset$
 5: $pathToGoal \leftarrow \emptyset$
    while $feasiblePaths = \emptyset$ do
      while $pathToGoal = \emptyset$ do
        $pathToGoal = Astar(L_c, 0)$ {Run A* with the
        given constraints, and no updates}
      end while
10:   Identify sleeve of $pathToGoal$ and its L-value
      Find SHP and identify anchor points $\{a_j\}$
      if feasibilityCheck() then
        find $feasiblePaths$ {pick a pair of ascent/descent
        paths that pass through all the anchor reachable set}
      end if
15: end while
```

The ascent path feasibility check remains much the same as that given by Abad-Manterola et al., but is described here as a separate function for clarity. For each anchor point on the SHP, we construct the anchor reachable set based on the terrain and the robot's capabilities. If robot can move so as to remove its tether from contact with the obstacle at that anchor point, that anchor point is considered passable. A series of passable anchor points means that the robot can reach a series of positions that allows it to ascend. If one of the anchor points is not passable, we add this sleeve's L-value to the list of constraints, and return to the main block of code to continue looking for feasible sleeves.

III. Impact of Newly Discovered Obstacles and Terrain Features

The algorithm reviewed in the previous section assumed a known, stationary map of the terrain prior to the planning process. In practice, at best a low resolution map will be available from a satellite flyover as a tethered robot such as Axel enters an extreme terrain. At worst, the only data available will be derived from on-board navigation cameras, and perhaps a mother ship's stereo camera mast. In either case, we assume that prior to entry into the extreme area, a nominal motion plan has been devised by applying the method of the previous section to the available map data. This section considers the following important question for online planning: how can the motion plan be properly updated to account for errors in the map that are inevitably encountered during the descent process? In particular, how can the path homotopy constraints be respected in light of these errors? Fortunately, the path planning procedure can be readily adapted to online path-planning.

Algorithm 2 feasibilityCheck()

```
    $continue \leftarrow True$
    $q \leftarrow g$
    $j \leftarrow length(\{a_j\})$
    $pathExists \leftarrow False$
 5: while $continue$ do
      construct $C_j(q)$ {Construct anchor reachable set}
      if $a_j$ is passable then
        $j \leftarrow j - 1$
        if $j = 0$ then
10:       $continue \leftarrow False$
          $pathExists \leftarrow True$ {feasible ascent path ex-
          ists}
        else
          $q \leftarrow$ SHP edge of $C_j(q)$
        end if
15:   else
        $L_c \leftarrow \{L_c, L\}$ {Add L-value to list of constraints}
        $continue \leftarrow False$
      end if
    end while
20: return $pathExists$
```

A. Adjusting the BTM

Assume that the robot has carried out the aforementioned offline planning, selecting a sleeve and its associated SHP, and an ascent/descent path pair. However, as it starts down the proscribed descent path, it receives new information, indicating that the original map is in error. A simple analysis shows that all of the possible errors fall into one or more of the following categories:

1) An entirely new, unforeseen, obstacle appears.
2) A map obstacle is found to be a phantom (i.e., the obstacle "disappears").
3) An obstacle has shifted location.
4) The boundaries of an obstacle or of the planning region are found to be different from those used to construct the preliminary plan.

Each of these changes will require a modification of the boundary triangulation of the descent slopes. Can this modification be handled in an online fashion? The answer to this question depends upon how changes in obstacle and terrain geometry affect the boundary triangulation. The classes of errors described above can be reduced to two simple classes of modifications: any error can be accommodated by adding and removing obstacles from the map–an existing obstacle or terrain boundary that has changed shape or position can simply be removed and replaced by a new, slightly different obstacle. Therefore, let us assume that we are changing BTMs only to reflect added or removed obstacles.

As proven in Lemma 1 below, the effects of bounded map errors can be localized to a small region, which we term the *affected region*. This region consists of all the BTM triangles that contain all or part of the added or removed obstacle components. Since the boundary of the affected

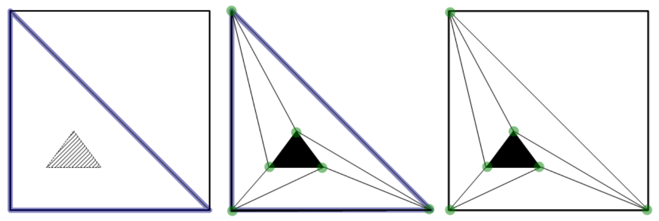

Fig. 9: Left: New obstacle added to BTM. Center: Triangulate affected area. Right: New BTM.

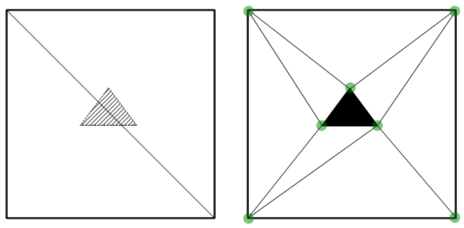

Fig. 10: Left: New obstacle added to BTM. Right: Retriangulate affected area for new BTM.

region consists of edges and vertices, which themselves must be boundary segments due to the properties of the original boundary triangulation, the affected area can be retriangulated as a BTM.

Lemma 1: Any boundary triangulated 2-manifold (BTM) can be locally retriangulated in the affected region around a new or removed obstacle to construct a proper boundary triangulation (see Definition 1) which seamlessly meshes with the boundary triangulation outside the affected region. The resulting triangulation is a BTM.

Proof: We assume that a new obstacle added to the map, or the obstacle to be removed from the map, is a bounded polygon. First consider the case of an added polygonal obstacle. This polygon will intersect a finite number of triangles in the boundary triangulation. The union of these triangles is termed the *affected region*, and is naturally a polygon. By the properties of boundary triangulation (Definition 1), the vertices bounding the affected region are boundary vertices. Hence, when a polygonal hole representing an obstacle is added to the BTM, a retriangulation of the affected region (incorporating the presence of the new obstacle) using de Berg's algorithm [10] results in a BTM of the affected region. Since the vertices of the affected area are, by definition, boundary vertices of the overall map, the retriangulation within the affected region maintains the BTM requirements for the overall map. Similarly, if an obstacle is removed, let the affected region consist of the union of the triangles whose edges or vertices are incident on the polygonal hole formed by the removed obstacle. Again, the boundary of the affected region satisfies the properties of a boundary triangulation, and thus the affected region can be retriangulated to form a BTM which naturally integrates with the existing BTM. □

Figure 9 illustrates how the process of adding an obstacle affects the triangulation. In the left panel, the new triangular obstacle added to the existing BTM lies wholly within a triangular simplex. That triangle is the affected region, and it is triangulated to form a local BTM (center diagram). Since all of the boundary vertices of this affected region are also boundary vertices of the larger map, the union of the retriangulation and the unaffected region forms a BTM (right panel of Fig. 9). The case of an obstacle falling into multiple simplices is illustrated in Figure 10. The two triangles incident to the obstacle become the affected region, and the entire square containing both simplices, as well as the newly discovered obstacle, are retriangulated (right-hand diagram of Fig. 10). As in Figure 9, the overall BTM remains a BTM, due to this specific retriangulation of the affected region.

The case for removing an obstacle is much the same, as can be seen by reading Figures 9 and 10 backwards. The affected region consists of the removed obstacle plus its adjacent simplices. The affected region is retriangulated to form a BTM, which conserves the BTM of the larger map.

B. Adjusting the Sleeve and Paths

Let the *current sleeve* denote the sleeve in which the robot's evolving descent path lies. When an unexpected obstacle is encountered, it may affect the geometry of the current sleeve, or it may not. In the most drastic effect, the current sleeve can be split into two sleeves, which will have coincident simplices up to the affected region. If there is no effect on the current sleeve, then the original plan can be continued until the next map error is encountered. The retriangulation may be useful later if the robot needs to backtrack.

If the newly observed obstacle intersects the current sleeve, it can have one of several effects:

1) The current sleeve is blocked. The robot must ascend until it finds a junction with an as-yet unexplored sleeve.
2) The sleeve geometry may be altered, but it does not alter the shortest homotopic path. In this case, there are no homotopy effects due to the map errors. The original plan will not be altered unless the obstacle blocks access to terrain whose trafficability is necessary for robot safety and controllability on the ascent or descent path.
3) The sleeve is not split into two pieces, but the shortest homotopic path is altered. The robot must reassess the feasibility of the new SHP. If the path is feasible, the robot can continue. Otherwise, the robot must backtrack to a junction with an as yet unexplored sleeve.
4) The current sleeve is split into two sleeves. The two sleeves will contain the same simplices up until the affected region. In the affected region, the new obstacle will force a *leftward* and *rightward* sleeve around the obstacle. The robot must then find and evaluate the feasiblility of the SHPs for each sleeve.

Similarly, removing an obstacle might free up a new sleeve, but it could also change the anchor points to make a path infeasible. Therefore, we must check for path intersection and feasibility. Algorithm 3 describes when to recompute sleeves and feasibility.

Algorithm 3 Online Path Planning Algorithm

```
    $updates \leftarrow \emptyset$
    repeat
      if $mapChanged$ then
        updateBTM()
5:      $updates \leftarrow \{updates, mapChanges\}$
        if affected area is in the current sleeve then
          $SHP_{old} \leftarrow SHP$
          update sleeve around SHP
          recompute SHP and anchor points
10:       if $SHP \neq SHP_{old}$ then
            if feasibilityCheck() $== False$ then
              $L_c = \{L_c, L\}$ {Add L-value to constraints}
              $Astar(L_c, update)$
              Backtrack to go down that sleeve
15:         end if
            select ascent and descent paths
          else
            if obstacle blocks ascent or descent paths or
            $L_{ascent} \neq L_{descent}$ then
              if feasibilityCheck() then
20:             select new ascent and descent paths
              else
                $L_c = \{L_c, L\}$
                $Astar(L_c, update)$
                Backtrack to go down that sleeve
25:           end if
            end if
          end if
        end if
      end if
30: until goal reached
```

Every map change invokes a local retriangulation–this is a computationally cheap operation. We ignore those changes that happen outside the current sleeve, placing them in a list of updates to be propagated when necessary. For those changes that affect the current sleeve, we update the sleeve to reflect the new triangulation, and recompute the SHP if necessary. If the SHP has changed, then the paths have been materially altered and we must replan from the affected region down to the goal. If there is no feasible path forward, this sleeve is marked as as infeasible by adding its L-value to the list of constraints. Similarly, if the SHP remains the same, but an obstacle blocks a path or comes between the ascent and descent paths, we must select new paths or block A* in that sleeve by adding that new L-value constraint. Note that L-values are only computed as A* expands that node.

We then run the A* search algorithm again, propagating map updates as we go. Having found a promising sleeve, the robot backtracks to the junction of the two sleeves, and starts down the new sleeve. If there is a feasible path, we must again select an ascent and descent path. By selecting a descent path within the set of feasible ascent paths, we ensure that the robot can always safely backtrack to its starting position.

This algorithm isn't optimal; because it doesn't recompute sleeves and paths globally, a removed or moved object may open up a more optimal sleeve, but the robot prefers to stay in the current sleeve when possible. However, it will eventually find a path to the bottom, if there is one.

C. Multiple Changed Obstacles

Numerous simultaneous updates to the map do not significantly change the online path planning algorithm. The updates can be processed in sequence by simply inserting a *for loop* before line 6 in Algorithm 3, with an end after line 28, running the intervening code for every affected region.

IV. Worked Example

This section covers an example of finding a feasible ascent and descent path pair using Algorithm 1. Then, we will demonstrate Algorithm 3 with an online example, in which a new obstacle appears in the sleeve of the descent path.

Assume that the robot starts with a map, which it can parse into a series of 2-dimensional tether-free and tether-demand planes with polygonal obstacles. Figure 11a shows an example tether-demand plane with two triangular obstacles shown in blue. The start and goal points are marked with X's, with $\mathbf{a_0}$ at the top, and $\mathbf{g}$ at the bottom. We start by triangulating this planar map into a BTM. The triangles outlined in black depict the BTM, and the orange line shows a potential path found with A*. Having found a path, one can reduce the crossing sequence to find its sleeve, shown in gray.

Using Hershberger and Snoeyink's algorithm [11], we find the SHP, shown in green in Figure 11b. Anchor points, where the SHP intersects the vertex of an obstacle, are circled in green. We examine each intermediate anchor point to determine if it is passable, and if all the anchor points in the list are passable, `feasibilityCheck()` returns `True`. In Figure 11c we have selected an ascent/descent path pair, shown in orange, from the set of all feasible paths.

After the robot completes its offline preplanning actions, it starts down the chosen descent path and turns a corner, only to discover a hitherto unknown obstacle. Figure 11d shows the path the robot has completed so far in red. Upon spying a new obstacle, the robot retriangulates the BTM in the affected area, and recomputes the SHP as in Figure 11e. The SHP change represents a change in the associated sleeve. Since the anchor points have changed, the robot must re-run Algorithm 2, `feasibilityCheck()`, to ensure that the sleeve is still traversable.

In our example, a feasible path exists, so the robot selects new ascent and descent paths and continues down the slope. Figure 11f shows the new descent path in orange, starting from the robot's current position. Its ascent path is not shown. The robot continues these steps until it reaches the goal. Every time a sleeve becomes untraversable, the search space is further constrained to avoid that L-value, and the robot

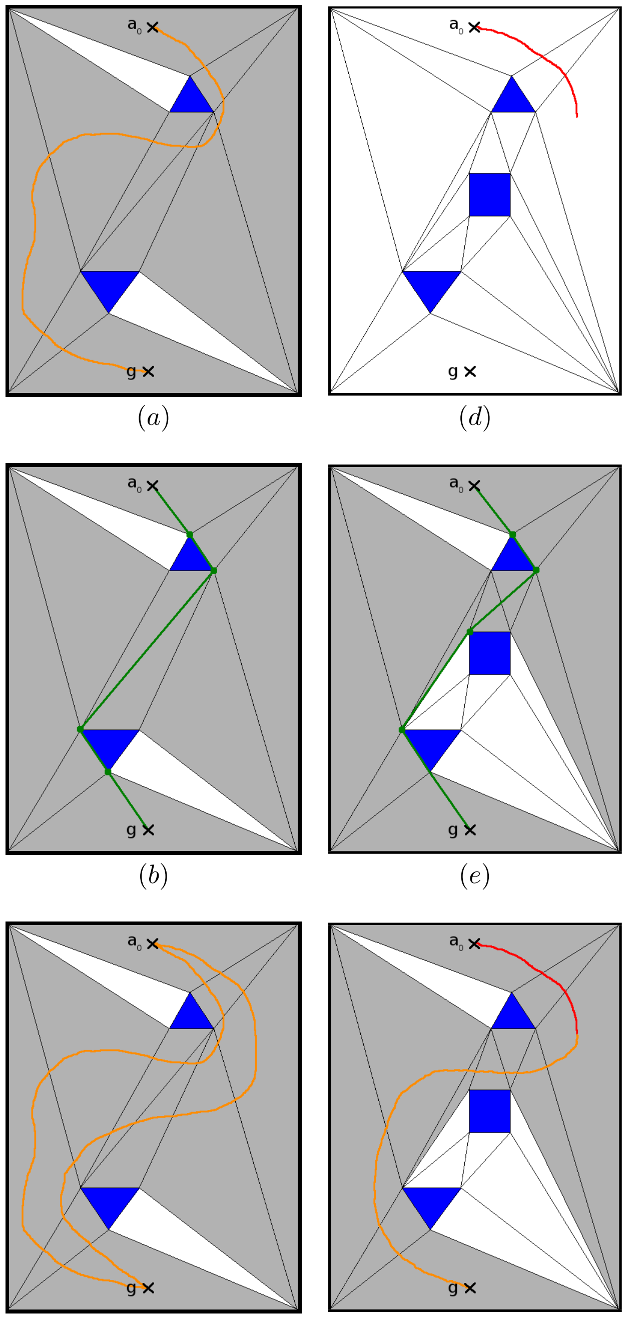

Fig. 11: An online example, with preplanning.

continues searching for new sleeves. Since it can backtrack, the robot can always return to the start to try another path in an as-yet unexplored sleeve.

V. Conclusion and Future Work

This paper presented an online path planning algorithm for a tethered extreme-terrain robot. Despite incomplete or incorrect initial information, our algorithm will find a safe path to the goal, if one exists. We have so far chosen to expand and compare only select sleeves and their SHPs, based on the heuristic in our A* algorithm. The sleeve is useful because it provides a useful framework for enumerating paths in a homotopy class. However, it is also possible to compute all SHPs by pruning the visibility graph of our map, as described in [17] and implemented in [18]. We intend to implement both methods in the future, to compare in practice the speed gain of precomputing SHPs versus the benefits of knowing the sleeve. We will implement an online planner in simulation and on the Axel rover.

Acknowledgments

The authors yould like to thank the Keck Institute for Space Studies for their support, and the reviewers for their insightful comments. The third author would like to acknowledge support from the JPL RTD program.

References

[1] P. Abad-Manterola, I. Nesnas, and J. Burdick, "Motion planning on steep terrain for the tethered Axel rover," in *Proc. IEEE Int. Conf. Robotics and Automation*, Shanghai, China, May 2010, pp. 4188–4195.

[2] Mars Science Laboratory: Wheels and legs. [Online]. Available: http://mars.jpl.nasa.gov/msl/mission/rover/wheelslegs/

[3] I. Nesnas, J. Matthews, P. Abad-Manterola, J. Burdick, J. Edlund, J. Morrison, R. Peters, M. Tanner, R. Miyake, B. S. Solish, and R. Anderson, "Axel and DuAxel rovers for sustainable exploration of extreme terrains," *J. Field Robotics*, pp. 663–685, Jul-Aug 2012.

[4] (2011) Mars Reconnaisance Orbiter: Multimedia. [Online]. Available: http://mars.jpl.nasa.gov/mro/multimedia/images/?ImageID=3580

[5] S. Hert and V. Lumelsky, "Moving multiple tethered robots between arbitrary configurations," *Int. Conf. Robots and Intelligent Systems*, vol. 2, pp. 280–285, August 1995.

[6] ——, "The ties that bind: Motion planning for multiple tethered robots," *Robotics and Autonomous Systems*, vol. 17, pp. 187–215, May 1996.

[7] ——, "Planar curve routing for tethered-robot motion planning," *Int. J. Computational Geometry and Applications*, vol. 8, pp. 437–466, August 1998.

[8] ——, "Motion planning in R3 for multiple tethered robots," *IEEE Trans. Robotics and Autonomation*, vol. 15, pp. 623–639, August 1999.

[9] P. G. Xavier, "Shortest path planning for a tethered robot or anchored cable," vol. 2, May 1999, pp. 1011–1017.

[10] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, 3rd ed. New York: Springer-Verlag, 2008.

[11] J. Hershberger and J. Snoeyink, "Computing minimum length paths of a given homotopy class," *Computational Geometry*, vol. 4, pp. 63–97, June 1994.

[12] B. Chazelle, "A theorem on polygon cutting with applications," in *23rd Ann. Symp. Foundations of Computer Science*, 1982, pp. 339–349.

[13] D. T. Lee and F. P. Preparata, "Euclidean shortest paths in the presence of rectilinear barriers," *Networks: An International Journal*, vol. 14, pp. 393–410, 1984.

[14] S. Bespamyatnikh, "Computing homotopic shortest paths in the plane," *Journal of Algorithms*, vol. 49, pp. 284–303, November 2003.

[15] S. Bhattacharya, M. Likhachev, and V. Kumar, "Search-based path planning with homotopy class constraints," in *Proc. 24th AAAI Conf. Artificial Intelligence*, Atlanta, Georgia, July 2010.

[16] S. Cabello, Y. Liu, A. Mantler, and J. Snoeyink, "Testing homotopy for paths in the plane," in *Proc. 18th Ann. Symp. Computational Geometry*, 2002, pp. 160–169.

[17] J. S. B. Mitchell, "Shortest paths and networks," in *CRC Handbook of Discrete and Computational Geometry*, J. E. Goodman and J. O'Rourke, Eds. CRC Press LLC, 1997, ch. 24, pp. 445–446.

[18] A. Akce and T. Bretl, "A compact representation of locally-shortest paths and its application to a human-robot interface," in *Proc. IEEE Int. Conf. Robotics and Automation*, Shanghai, China, May 2011.