Motion Planning for Climbing Mobility with Implementation on a Wall-Climbing Robot

Keenan Albee
Space Systems Laboratory
Massachusetts Institute of Technology
albee@mit.edu

Antonio Terán Espinoza
Space Systems Laboratory
Massachusetts Institute of Technology
teran@mit.edu

Kristina Andreyeva
Columbia University
ka2487@columbia.edu

Nathan Werner
Columbia University
nmw2120@columbia.edu

Howei Chen
Columbia University
hlc2128@columbia.edu

Tamas Sarvary
Columbia University
tms2172@columbia.edu

***Abstract*—Future autonomous planetary explorers will require extreme terrain mobility to reach areas of interest, such as walled lunar pits and steep Martian rock layers. Climbing mobility systems are one proposed answer, requiring efficient and kinematically feasible motion planning for autonomous operation. Similarly, climbing planning is applicable to other microgravity situations requiring constant end effector contact with discrete handholds. This paper proposes a planning framework that poses kinematic climbing planning as a discrete optimal planning problem. Motion primitives are used to encourage large robot body workspaces and beneficial connections between climbing stances. A wall-climbing planner simulation is presented, along with implementation on a hardware demonstration testbed that successfully recognized, navigated, and climbed an arbitrary vertical wall.**

TABLE OF CONTENTS

1. INTRODUCTION

Planetary surface robotics has enabled exploration of the solar system to go mobile. While lander probes can provide data at a single sample site and orbital surveyors can make observations at a distance, mobile surface robotics provides access to terrains and areas otherwise unreachable. Particularly on Mars, where most recently the MER and MSL missions have seen great success [3], robotic planetary explorers expand the horizons of science-gathering.

These robots must operate in sites that meet their operational intent, which to this day has been limited to rocker-bogie surface rover systems; Parness provides a summary of wheeled exploration approaches and some of their limitations [15]. Interesting scientific discoveries may await in terrain that is inaccessible by traditional rover systems: a few examples include steep-walled lunar pits, sharp crater walls, and layered Martian rock layers [15] [1]. The NASA Technology Roadmaps, which guide technology development to aid in future missions, identify Extreme Terrain Mobility (TABS 4.2.1) as a key technology for in-situ access to these new and exciting sample sites. Specifically, climbing mobility systems are mentioned as an area for development [1].

978-1-5386-6854-2/19/$31.00 ©2019 IEEE

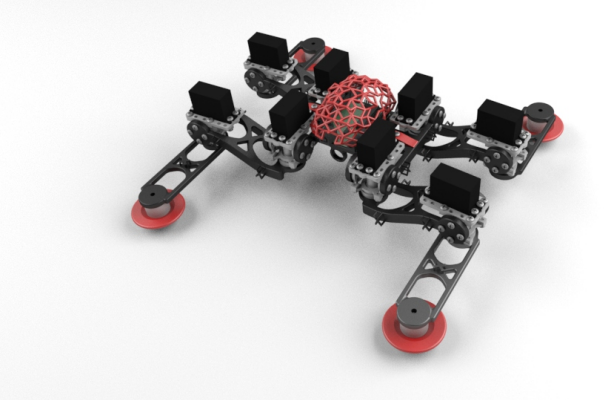

Figure 1. A CAD rendering of WALLY, the hardware testbed.

As is often the case in planetary surface and microgravity robotics, significant autonomy is needed to deal with large communications delays, challenging environments, and complex robotic systems. Pavone et al. summarize some of the key autonomy challenges for space exploration, including extreme terrain mobility [16]. This work proposes an algorithm suited for guiding a limbed climbing robot in real-time. The novel aspect comes in the explicit formulation of this variety of climbing problem, and posing it as a discrete optimization well-suited for kinematic motion primitives. Finally, the approach is demonstrated on a custom hardware testbed, WALLY, and its accompanying fully-fledged simulation environment. The use of such a system could fling open new doors for planetary explorers desiring steep terrain access, enabling new scientific discovery.

The paper is structured in the following format: Section 2 includes background knowledge and previous work related to planetary climbing robotics and limbed extreme terrain motion planning; the problem formulation is described in Section 3; the development of the proposed algorithmic approach is presented in Section 4; hardware implementation and results are shown in Section 5; and the conclusion is presented in Section 6.

2. BACKGROUND AND RELATED WORK

Common strategies and techniques relevant to the planning framework proposed in Section 4 are briefly discussed in this section. Subsequently, related work pertaining to the motion planning field is presented.

Background Theory

A method for finding optimal solutions for discrete problems in which a sequence of decisions needs to be chosen – such as discrete motion planning problems – is to explore the available decision space and build a corresponding search tree [6].

Amongst the existing strategies for exploring such a search tree in an informed manner, **A* search** is one of the most widely known [17]. **A*** assigns a cost J to a tree node n that is the combination of the cost-to-come to said node, expressed as $g(n)$, and the cost-to-go from that node n to the goal:

$$J(n) = g(n) + h(n). \tag{1}$$

Thus, if the objective is to find the cheapest, or optimal, way for reaching a solution, a reasonable strategy is to explore the tree by choosing the option with the lowest cost. The function $h(\cdot)$ is called a heuristic, and provided that it satisfies some conditions (admissible), the aforementioned search strategy is provably complete and optimal [17].

Related Work

Limbed extreme terrain mobility involves the search for a feasible path from a start state to a goal state over a set of discrete end effector placement locations. Depending on the source in the literature, these locations are commonly referred to as footholds, footsteps, footfalls, or handholds; in this paper, the term *handholds* is chosen, since it appropriately fits the climbing problem.

A common reference for climbing mobility work is [5], in which the extreme terrain mobility problem is separated into two components: determining these handholds, often subject to kinematic or dynamic constraints, and performing an efficient search over these holds.

Alternatively, other approaches to the problem first focus on developing a path and afterwards in finding the handholds that fit this trajectory: this approach is more common with gaited systems, that are not as limited in their handhold availability as the climbing problem may be.

An additional distinction between motion planning problems is the quasi-static versus the kinodynamic case: quasi-static problems ignore the system dynamics (i.e., don't engage in jumping or bounding motions), while dynamic problems perform trajectory planning. The quasi-static assumption is very common for climbing problems, since climbing systems are often in a stable hanging configuration and make careful stance transitions. This overlap between climbing and footstep planning problems seems very large, and a fruitful area for future merging of the two disciplines.

In terms of the footstep planning problem, Zucker provides a succinct summary, and proposes a handhold planner followed by a trajectory optimization step, looking at the dynamic case [21]. Other recent sophisticated planners, often for legged locomotion, deploy sampling-based approaches with stability guarantees later added [19].

On the other hand, focusing on the climbing problem – specifically the extreme terrain mobility problem in the quasi-static sense – a variety of distinct approaches have been mentioned. In terms of climbing robots, many works focus mainly, or almost exclusively, on the mechanical design [2], [18], [7]. However, the warranted planning algorithms for steering these systems require attention.

The climbing problem is posed in detail by Bretl, who initially proposes a one-step planning strategy for exploring a hybrid search space using a three-limbed climbing robot based on heuristic methods – making use of kinematic constraints and rough discrete plans – in order to address the intractability imposed by the need to perform online search at each time step [4]. The same authors later explore an approach based on probabilistic roadmaps (PRM) as a multi-step planning framework, reinforcing the idea of initially pruning the search space through coarse proposals to afterwards perform finer one-step adjustment between stances, making online search feasible [5].

More recently, platforms have been developed demonstrating quasi-static walking and climbing robots. An example of this is the LEMUR 3 robot [15], which was developed with a very similar intent in mind, but for which the literature has so far focused mainly on the mechanical aspects of the design; LEMUR has a strong history of development at NASA JPL, and some of Bretl's work was implemented on early versions of this system [11]. Additionally, JPL also developed RoboSimian for the DARPA Robotics challenge[10], which includes a sophisticated quasi-static path planner working under the assupmtion that footstep selection from a widely available world is available.

Additional works focus on gaited motion whose presented techniques do not apply to cases in which limited and discrete handholds are involved [18].

The scenario in which a quasi-static robot needs to traverse a world with very limited handholds available could benefit from efficient, real-time planning, yielding an area for which development of efficient motion planning algorithms is warranted. This problem is of particular interest to the space robotics community, as evidenced by recent projects [15][10], because of the implications for robotic planetary exploration and mobility along microgravity structures. The problem formulation for such cases is posed in the next section, and a potential real-time solution is presented in Section 4.

3. PROBLEM FORMULATION

The two-dimensional world $\mathbf{W}$ in which the climbing robot operates is defined as

$$\mathbf{W} = \{(x, y) : w_{l_x} \leq x \leq w_{r_x}, w_{l_y} \leq y \leq w_{r_y}\},$$

where $\mathbf{w}_l$ and $\mathbf{w}_r$ denote the bottom-left and top-right vertices of this 2D world. Within this rectangle, a set $\mathbf{H} \subset \mathbf{W}$ of holds of the form

$$\mathbf{H} = \{\mathbf{h}_i = (x_i, y_i) \in \mathbf{W}\}$$

is defined; these holds $\mathbf{h}_i$ denote the positions within this two-dimensional space in which it is feasible for the robot to place its limbs' end effectors. A pictorial representation is shown in Figure 2.

The initial conditions of the climbing robot are thus expressed as the combination of the robot limbs' end effector positions

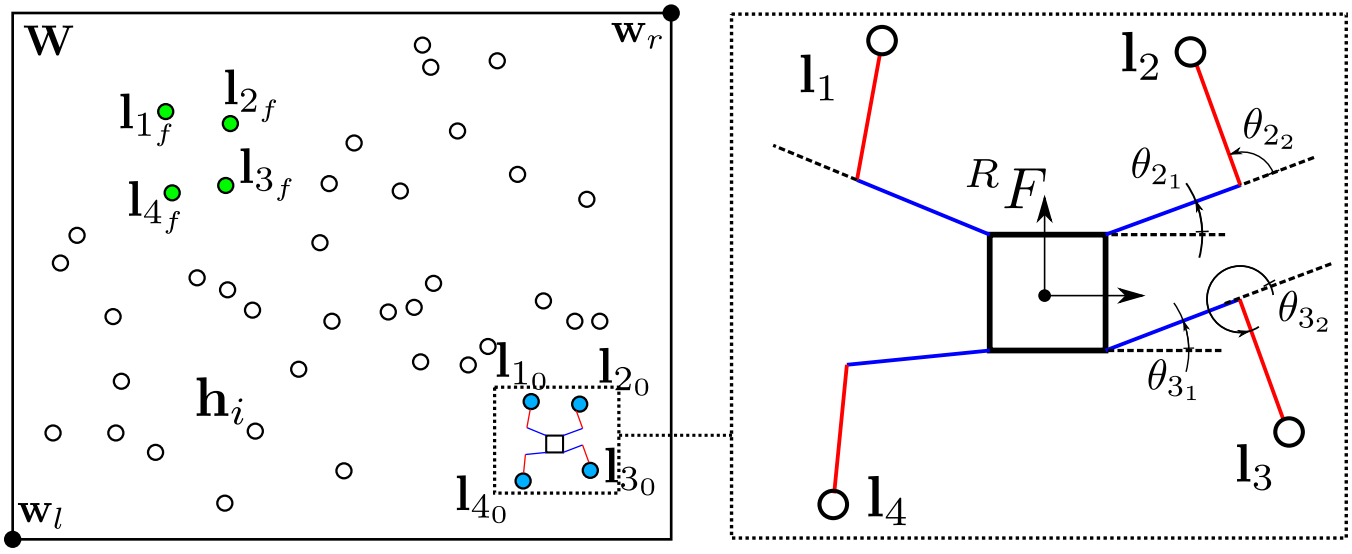

Figure 2. Sample scenario showing a robot with $N = 4$ limbs and $M = 2$ joints. (left) A pictorial depiction of the two-dimensional world $\mathbf{W}$ is shown; the hollow circles denote the available holds $\mathbf{h}_i \in \mathbf{H}$, with the sets of holds corresponding to the initial (blue) and final (green) conditions highlighted. (right) A zoomed image of the robot is presented, in which the stance's parameters are explicitly represented.

$\mathbf{l}_{k_0}$ and their corresponding joint angles θ_{k_j}, forming a *stance*

$$\mathbf{R}_0 = \{(\mathbf{l}_{k_0}, \boldsymbol{\theta}_{k_0}) : \mathbf{l}_{k_0} \in \mathbf{H}, \forall k \in [1, \ldots, N]\},$$

where N denotes the number of limbs, $\boldsymbol{\theta}_{k_0}$ the set of angles θ_{k_j} corresponding to the initial j-th joint angle of the k-th limb $\forall j \in [1, \ldots, M]$, and M denoting the number of joints for the corresponding limb. A description of this configuration is similarly shown in Figure 2.

The objective of such a scenario is to determine an incremental sequence of stances $\mathbf{R}_{0:F}$ such that the robot is able to traverse from its initial conditions to a desired set of final holds $\mathbf{L}_f$; for the case shown in Figure 2, $\mathbf{L}_f = \{\mathbf{l}_{1_f}, \mathbf{l}_{2_f}, \mathbf{l}_{3_f}, \mathbf{l}_{4_f}\}$, which is shown in green.

For a stance to be valid, the inverse kinematics of the parallel robotic mechanism must be obeyed. That is, for a given stance $\mathbf{R}_i$, the following constraints must be satisfied:

- for the corresponding set of end effector holds $\mathbf{L}_i$, the corresponding set of angles $\boldsymbol{\theta}_i$ must satisfy the robot's inverse kinematics;
- the robot links must not self-collide, collide with other links, collide with other geometric objects (e.g., the base), or exceed actuator angle limits.

The problem posed is then that of finding an optimal path from a starting stance $\mathbf{R}_0$ to a goal stance $\mathbf{R}_f$, subject to the aforementioned constraints. This can be concisely expressed as a discrete time optimization problem of the form

$$\begin{aligned} \underset{\mathbf{u}}{\text{minimize}} \quad & \sum_{i=0}^{F-2} ||m(\mathbf{R}_i) - m(f(\mathbf{R}_i, \mathbf{u}_i))||_2 \\ \text{subject to} \quad & \mathbf{R}_{i+1} = f(\mathbf{R}_i, \mathbf{u}_i), \\ & \mathbf{R}_F = \mathbf{R}_f, \\ & \mathbf{R}_i \in \mathcal{R}_{\text{feasible}}, \\ & \mathbf{u}_i \in \mathcal{U}, \end{aligned}$$

where $\mathcal{R}_{\text{feasible}}$ is the available stance space spanned by the kinematic constraints of the robot, $\mathcal{U}$ is the set of allowable actions, F is the length of the calculated stance sequence, $f(\cdot)$ returns the corresponding acceptable stance after choosing control action $\mathbf{u}$, and $m(\cdot)$ returns the 2D position of the robot's fixed body frame origin given a specified stance. The cost function can of course be modified to meet a goal other than minimum distance traveled.

4. Approach

The algorithmic solution proposed herein is suited for multi-limbed planar robotic explorers. For the presented analysis, a four-limbed 4RRR mechanism, such as the one shown in Figure 2, is used. In [13] a similar mechanism is proposed, but only the local planning problem is discussed. The described formulation is posed as a planning problem, with the **A*** strategy being the chosen technique to tackle the optimization over a discrete sequence of stances. In order to create connections between the stances at adjacent time steps, a local planner is paired with a predefined set of allowable stance configurations; this reduction of the allowable workspace is performed in order to render the problem tractable and to aid in the selection of stances, tantamount to the motion primitives commonly found in the planning literature [9]. In addition, different **A*** heuristics are also proposed to promote faster solution times.

In this section, the robot's configuration and state space are first shown, followed by a description of its forward and inverse kinematics. Subsequently, the **A*** global planner and the algorithmic solution to the optimization problem is presented in detail.

Configuration Space and Holds

The kinematic state space of the robot system is described by an 11-dimensional vector formed by the eight joint angles and the three components of the 2D transformation that describe the robot's fixed body frame ${}^{\mathbf{R}}F$ with respect to the world

frame ${}^{\mathbf{W}}F$. Thus, the state space vector $\mathbf{x}$ can be expressed as

$$\mathbf{x}^\top = \begin{bmatrix} T_{\mathbf{W}}^{\mathbf{R}\top} & \boldsymbol{\theta}^\top \end{bmatrix}, \text{ with}$$

$$T_{\mathbf{W}}^{\mathbf{R}} = \begin{bmatrix} {}_w x & {}_w y & \theta_{\mathbf{W}}^{\mathbf{R}} \end{bmatrix}^\top$$

$$\boldsymbol{\theta} = [\theta_{1_1} \;\; \theta_{1_2} \;\; \theta_{2_1} \;\; \theta_{2_2} \;\; \theta_{3_1} \;\; \theta_{3_2} \;\; \theta_{4_1} \;\; \theta_{4_2}]^\top.$$

Geometry and Forward/Inverse Kinematics

The kinematic constraints of the 4RRR robot are governed by the forward and inverse kinematics of the robotic manipulator considered here. Notably, a climbing robot can be thought of as transitioning between being a serial and parallel mechanism. When the base is moving with end effectors attached to holds, the system is the parallel 4RRR mechanism mentioned above. However, when an individual end effector is released and moves independently, it effectively acts as a serial manipulator, with a new predetermined pose for its base. Therefore, two sets of forward and inverse kinematics are needed: one for the 4RRR parallel mechanism, and one for the RR planar mechanism.

RR planar The RR planar mechanism is a typical example for forward and inverse kinematics, with its trigonometric derivation readily available in many references [20]. The forward kinematics are expressed as

$$\mathbf{x}_{EE_j} = \begin{bmatrix} x_j \\ y_j \end{bmatrix} = \begin{bmatrix} l_1 \cos(\theta_{j_1}) + l_2 \cos(\theta_{j_1} + \theta_{j_2}) \\ l_1 \sin(\theta_{j_1}) + l_2 \sin(\theta_{j_1} + \theta_{j_2}) \end{bmatrix} + \begin{bmatrix} x_{j_0} \\ y_{j_0} \end{bmatrix}$$

with the pairs (l_{j_1}, θ_{j_1}) and (l_{j_2}, θ_{j_2}) corresponding to the link length and joint angle of the first and second joints of the j-th, limb respectively. x_{j_0} and y_{j_0} correspond to the limb base in the robot fixed frame.

4RRR planar The 4RRR planar mechanism, if its base is held at a constant $\theta_{\mathbf{W}}^{\mathbf{R}}$, can be thought of as the equivalent of finding the inverse kinematics of each individual RR serial mechanism. The inverse kinematics of the system's j-th limb are thus expressed as

$$\theta_{j_2} = \operatorname{atan2}\left(-\sqrt{(1 - D_j^2)}, D_j\right)$$

$$\theta_{j_1} = \operatorname{atan2}(y_j, x_j) - \arctan\left(\frac{l_{j_2}\sin(\theta_{j_2})}{l_{j_1} + l_{j_2}\cos(\theta_{j_2})}\right),$$

for all angles in $\boldsymbol{\theta}$, where $\mathbf{x}_j = (x_j, y_j)$ are the end effector positions, $\mathbf{l}_{\mathbf{j_1}} = (l_{j_1}, l_{j_2})$ the lengths of the first and second links, respectively, with the parameter $D_j = (||\mathbf{x}_j||_2^2 - ||\mathbf{l}_{\mathbf{j_1}}||_2^2)/(2 l_{j_1} l_{j_2})$.

A* *Global Planner*

If the robot is quasi-static (i.e., not jumping), the problem can be viewed as connecting pairs of stances $(\mathbf{R}_i, \mathbf{R}_{i+1})$ while satisfying the kinematic constraints. Moving from $\mathbf{R}_i$ to $\mathbf{R}_{i+1}$, however, is a potentially perilous problem, since the number of available options is virtually unlimited (e.g., joint angles are variables in a continuous domain). To choose an $\mathbf{R}_{i+1}$, a set of four $\mathbf{h}_i$ must also be selected that are valid for the given robot geometry; further, $\boldsymbol{\theta}_i$ must be specified. The options for $\mathbf{h}_i$ then grow as $\mathcal{O}(n^4)$, where n is $|\mathbf{H}|$, i.e., the total number of holds in the world $\mathbf{W}$.

The global **A*** planner is summarized in Algorithm 1, which is outlined here. Note that this assumes that $h(n)$ is an admissible heuristic. Central to the planner is the idea that the set of potential next stances for any given node can be discretized into some sensible motions. This is similar to the concept of motion primitives, whereby motion types for a trajectory are discretized into some common, sensible types [12]. For this particular manipulator configuration, a set of eight stance primitives to inform the robot's next stance was generated.

Algorithm 1 Global Planner Logic.

procedure CLIMB-A*($\mathbf{R}_0, \mathbf{R}_f$)
  $\mathcal{R}_{\text{explored}} \leftarrow \{\emptyset\}$
  $\mathcal{R}_{\text{open}} \leftarrow \mathbf{R}_0$
  while $\mathcal{R}_{\text{open}} \neq \{\emptyset\}$ **do**
    $\mathbf{R}_{cur} \leftarrow$ `findMin`($\mathcal{R}_{\text{open}}$)
    if $\mathbf{R}_{cur} == \mathbf{R}_f$ **then**
      return `path`($\mathbf{R}_{cur}$)
    $\mathcal{R}_{\text{children}} \leftarrow$ `GenerateChildR`($\mathbf{R}_{cur}$)
    $\mathcal{R}_{\text{explored}} \leftarrow \mathcal{R}_{\text{explored}} \cup \mathbf{R}_{cur}$
    for all $\mathbf{R}_{child} \in \mathcal{R}_{\text{children}}$ **do**
      if $\mathbf{R}_{child} \in \mathcal{R}_{\text{explored}}$ **then**
        `continue`
      $g_{child} \leftarrow g(\mathbf{R}_0, \mathbf{R}_{cur}) + s(\mathbf{R}_{cur}, \mathbf{R}_{child})$
      $h_{child} \leftarrow h_T(\mathbf{R}_{child})$
      if $\mathbf{R}_{child} \notin \mathcal{R}_{\text{open}}$ **then**
        $\mathcal{R}_{\text{open}} \leftarrow \mathcal{R}_{\text{open}} \cup \{\mathbf{R}_{child}\}$
      else if $g_{child} \geq (g_{child} \in \mathcal{R}_{\text{open}})$ **then**
        `continue`
      $(\mathbf{R}_{child} \in \mathcal{R}_{\text{open}}) \leftarrow$ `parent`(R_{child})
      $(\mathbf{R}_{child} \in \mathcal{R}_{\text{open}}) \leftarrow g_{child}$
      $(\mathbf{R}_{child} \in \mathcal{R}_{\text{open}}) \leftarrow h_{child}$
  if $\mathbf{R}_{cur} != \mathbf{R}_f$ **then**
    return `error`

Stance primitives reduce the branching factor of each node in the search. The chosen stance primitives are shown in Figure 3; as is shown in the figure, these configurations are pre-selected in order to produce large, usable workspaces. For instance, an upward climbing motion will seek a stance primitive that extends its workspace far downward in order to connect to a corresponding upward motion. In effect, this makes efficient use of end effector positioning so that workspaces are large and well-intentioned, avoiding wasteful short motions of the base as much as possible. The cost for moving between stance $\mathbf{R}_i$ to $\mathbf{R}_{i+1}$ is given by the function $s(\cdot, \cdot)$, defined as the displacement between the robot's center of geometry at those two stances, or

$$s(\mathbf{R}_i, \mathbf{R}_{i+1}) = \left|\left|\left[T_{\mathbf{W}}^{\mathbf{R}_i}\right]_{1,2} - \left[T_{\mathbf{W}}^{\mathbf{R}_{i+1}}\right]_{1,2}\right|\right|_2.$$

The eight stance primitives are used for eight possible directions of travel, $d \in \mathcal{D}$. These directions of travel are the approximate maximum linear motion of $T_{\mathbf{W}}^{\mathbf{R}_i}$ without violating the workspace constraints of $\mathbf{R}_i$. This is checked using the procedure `moveMax`, which simply takes evenly spaced samples along the sample direction until the workspace is

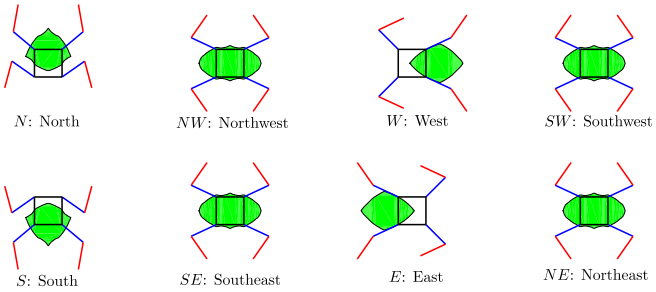

Figure 3. The stance primitives representing desirable positions of θ in order to produce large workspaces, in green. From top left and moving clockwise, the $d \in \mathcal{D}$ are: up, up-left, left, down-left, right-up, right, right-down, down.

violated. The next largest motion is selected, resulting in the subsequent position $({}_w x_{i+1}, {}_w y_{i+1})$. Thus, the call to the procedure `generateChildR`$(\mathbf{R}_i)$ limits the maximum number of child stances.

However, this $({}_w x_{i+1}, {}_w y_{i+1})$ has many potential handhold $\mathbf{h}_{i+1}$ combinations that are kinematically feasible, and it is at exactly this point where the stance primitives are useful in order to make an informed decision for choosing an optimal subset. The procedure `findHolds` selects the $\mathbf{h}_{i+1}$ with the smallest deviation from that of the stance primitive, and selects these holds, with the aim of providing a similar stance with a similarly well-intentioned workspace. Because exactly positioned holds might not be available, an inverse kinematics check must be performed to verify that the workspace delimited by $\mathbf{h}_{i+1}$ actually does include $({}_w x_{i+1}, {}_w y_{i+1})$. If so, the $\mathbf{R}_{i+1}$ is added as a child. A sample workspace and set of maximum translation points with respect to the robot's center of geometry is depicted in Figure 4.

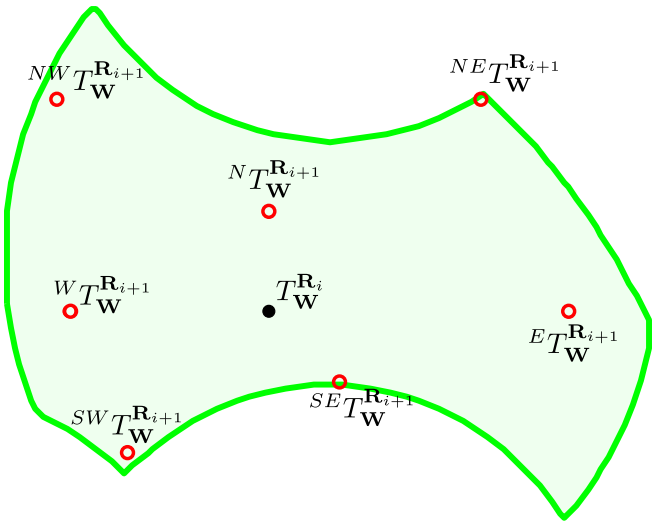

Figure 4. A sample workspace created from a stance $\mathbf{R}_i$. The central black point is $T_{\mathbf{W}}^{\mathbf{R}_i}$, while accompanying red points are the maximum motions along $d \in \mathcal{D}$ found by `moveMax`. Note that ${}^S T_{\mathbf{W}}^{\mathbf{R}_{i+1}}$ is not shown since it does not meet minimum requirements on motion of the base.

This child generation procedure creates at most 8 children along the aforementioned pre-specified set of directions $\mathcal{D}$. While this limits optimality, it makes the problem discrete and tractable by performing motions that approximate well-intentioned moves. Thus, a local planner that makes use of this procedure, which is outlined in Algorithm 2, is able to generate forward linear plans between $\mathbf{R}_i$ and $\mathbf{R}_{i+1}$.

Algorithm 2 Local Planner Logic

```
1: procedure generateChildR*(R_i)
2:     R_children ← {∅}
3:     for all d ∈ D do
4:         (_w x_{i+1}, _w y_{i+1}) ← moveMax(R_i, d)
5:         h_{i+1} ← findHolds(x, y, d)
6:         if R_{i+1} ∈ R_feasible then
7:             R_children ← R_children ∪ R_{i+1}
8:     return R_children
```

The optimization cost J is then approximated – as is standard for an **A*** planner – by two distinct functions: $g(\mathbf{x})$ as the cost-to-come, and $h(\mathbf{x})$ as the cost-to-go, rendering the cost as $J = g(\mathbf{x}) + h(\mathbf{x})$. Ideally, $h(\mathbf{x})$ is an exact measure of the cost-to-go, but in practice heuristics must be used to make the problem tractable. Multiple heuristics were employed for the proposed approach, with various weightings applied with results evaluated qualitatively based on simulation solution times and optimality. Euclidean distance to the goal is a common under-approximation yielding an admissible heuristic:

$$h_1(\mathbf{x}_i) = \left\| \left[T_{\mathbf{W}}^{\mathbf{R}_i} \right]_{1,2} - \left[T_{\mathbf{W}}^{\mathbf{R}_f} \right]_{1,2} \right\|_2,$$

where $[\cdot]_{1,2}$ denote the first two elements of the argument (i.e., position of the robot's center of geometry).

Furthermore, a slight modification of the original optimization problem seeks not only to minimize the distance traveled, but also to promote time optimality. The minimum $T_{\mathbf{W}}^{\mathbf{R}}$ path might consist of many small body motions, which require significant time to reposition limbs to the new stance. It is possible that paths which traverse a longer overall $T_{\mathbf{W}}^{\mathbf{R}}$ distance may actually be faster, since they require cycling between fewer stances. The following two heuristics were proposed to address this concern:

$$h_2(\mathbf{x}_i) = \left\| \left[T_{\mathbf{W}}^{\mathbf{R}_i} \right]_{1,2} - \left[T_{\mathbf{W}}^{\mathbf{R}_{i+1}} \right]_{1,2} \right\|_2, \text{ and } h_3(\mathbf{x}_i) = i$$

However, neither heuristic was used significantly in actual hardware testing, as it was found that solution time could easily increase if, for instance, emphasis on longer $T_{\mathbf{W}}^{\mathbf{R}}$ motion unnecessarily promoted search deviating from $\mathcal{R}_{\mathcal{G}}$, which is the set of goal stances.

Finally, an additional heuristic in favor of decreasing solution time is proposed, which accounts for distance already traveled by the torso in order to encourage depth of the search tree rather than breadth, at the cost of optimality, expressed as

$$h_4(\mathbf{x}_i) = \sum_{j=1}^{i} \left\| \left[T_{\mathbf{W}}^{\mathbf{R}_j} \right]_{1,2} - \left[T_{\mathbf{W}}^{\mathbf{R}_{j-1}} \right]_{1,2} \right\|_2.$$

These aforementioned metrics are combined together to form $h_T(x_i)$ using a weighting vector $\mathbf{w}$, obtaining an expression

of the form

$$h_T(\mathbf{x}_i) = \sum_{j=1}^{4} w_j \cdot h_j(\mathbf{x}_i),$$

where the weighting factor w_j corresponds to the importance of the metric expressed by the j-th heuristic $h_j(\cdot)$.

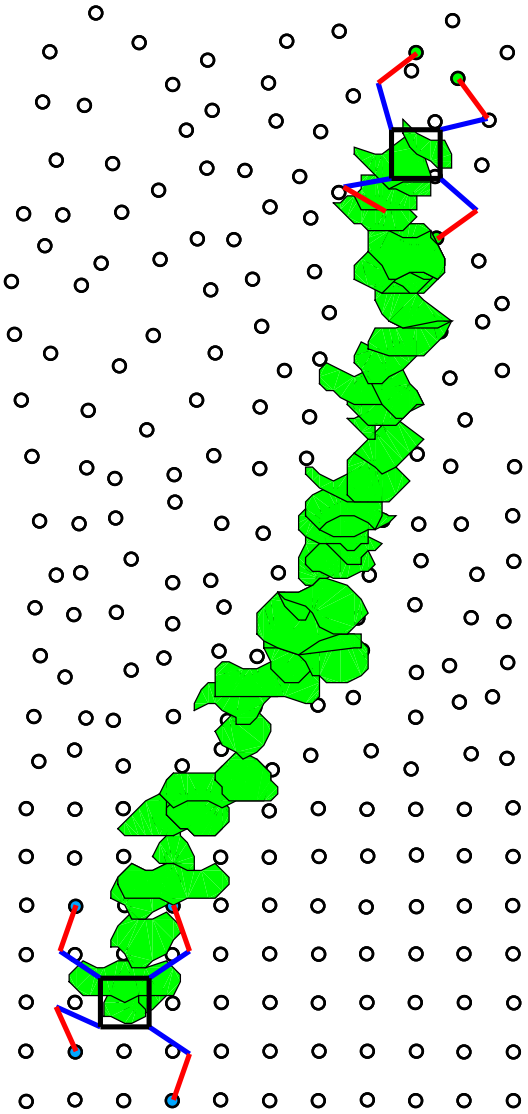

Figure 5. A series of connected workspaces for the solution to a climb. Notice how workspaces differ in shape based on the desired type of motion at each $\mathbf{R}_i$.

5. Results and Discussion

A hardware demonstrator and accompanying simulator were developed, described in further detail in the *Hardware Implementation* and *Planning Simulation* sections. A 4' x 8' test wall with increasingly sparse and randomized handholds was successfully traversed from bottom to top, shown in Figure 8. Solutions times of approximately $4\ s$ for climbs from the bottom left of the assembled test wall were obtained. A basic computer vision system was developed, using built-in MATLAB circle-finding functions for holds and fiducials on the robot. Some tweaking of the computer vision system was necessary: while some climbs were possible with full estimates of the world from an offset camera, a manually measured version of the wall was also provided in lieu of hold estimates to ensure accuracy.

Planning Simulation

A MATLAB simulation of the proposed algorithm was developed to allow for testing of algorithm modifications and for hardware implementation. The simulation is filled with many features for better understanding the problem, including visualization and plotting tools and video playback of motion plans. MATLAB's object-oriented programming capability encapsulates the robot and its motion planning capability, along with a separate wall object for randomly generating worlds, $\mathbf{W}$. The simulation is highly customizable, allowing for modification of the key robot geometry including link lengths, joint angle centering and limits, wall geometry (with pre-generated walls and randomly-generated varieties), initial and final conditions, and limb orientation (i.e. one of the two solutions provided by the inverse kinematics). Some features that are customizable but not immediately accessible from the main configuration file are specifying stance primitives and heuristic weightings.

A graphical user interface (GUI), shown in Figure 6, was created for ease of simulation operation if fewer parameters are desired for tweaking. The GUI provides a view of the image read in by the robot system, the generated world, and options for setting joints angles, turning end effectors on and off, and performing some preset motions. The goal may be selected at left, and the ensuing planning and playback is shown at right, with plan execution controlled using a playback interface at center.

Heuristics were evaluated qualitatively by searching for a compromise between solution time, time optimality, and distance optimality, in decreasing order of importance. A final weighting that was rather successful during hardware testing was $\mathbf{w} = [0.7,\ 0,\ 0,\ 0.3]^\top$.

Solution times were evaluated on a randomly generated wall with hold density of $.11\ holds/unit^2$, l_i of length $4\ units$ and starting and ending in the stances shown in Figure 5. Note that for runtime tests, $\mathbf{W}$ was randomly generated, as opposed to the tiered $\mathbf{W}$ shown in Figure 5. The average runtime for this test scenario was $2.35\ s$, with an average number of $43\ stances$ in the solution path and an average of $104\ nodes$ expanded, over 100 randomly generated $\mathbf{W}$. The simulation was performed on a 16 GB, 8 core Intel i7-4700MQ CPU @ 2.40GHz, running Linux Mint 18.3. Note that these MATLAB simulation runtimes could further benefit from significant computational speedup using a more efficient computational implementation.

The algorithm was highly successful at achieving solutions in the limited handhold, obstacle-free environment while satisfying the kinematic constraints of the mechanism in computation time feasible for real-time use, particularly if short plans are desired. One drawback of the motion primitive approach is the loss of optimality from downselecting actions: for instance, when holds become sparse, there is a chance that no motion primitive will yield an appropriate stance and so a feasible solution may be skipped. In addition, the runtime of the algorithm increases significantly if many sparse regions exist in an otherwise handhold-dense $\mathbf{W}$, akin to having many obstacles. This situation requires some reconsideration of the algorithm, perhaps heuristics to account for obstacle avoidance or blending with sampling-based planning to break

out of obstacle regions.

Further, a nearest neighbor search is performed for selecting the holds closest to a desired hold. This is a potential bottleneck, though for reasonable hold densities it is not a major concern. One difficulty observed during simulation was the occasional failure of `GenerateChildR` to identify if a direction d was not achievable by following a simple linear path. Because the 4RRR workspace is not necessarily convex, a linear translation might actually leave the workspace, violating kinematic constraints. While not a common occurrence, future work will need to address this local planning problem in more detail.

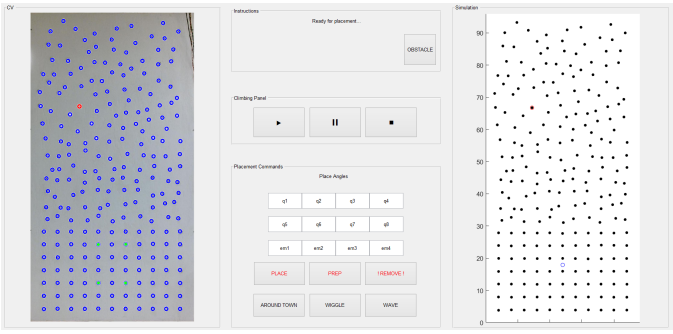

Figure 6. The GUI for simulation use, showing a captured or presupplied wall image on the left and corresponding simulation visualization on the right.

A more sophisticated `moveMax` would be beneficial, so that the absolute furthest point within a tolerance could be achieved, rather than the rough linear sampling currently performed along members of $\mathcal{D}$. Automated generation of stance primitives, for example, generating those with the largest workspaces in the most desirable locations is an area of promising future work, and would blend well with adapting the algorithm to different robotic mechanisms and those with differing kinematic constraints.

Finally, assuming a rigid end effector grip is one of the major simplifications made in this problem. Robot stance equilibrium constraints are explored extensively by Bretl and others looking at automated rock climbing robotics. The additional constraints imposed by requiring good grasp configurations and avoiding undesirable or dynamically unfeasible stances would be a worthwhile and necessary extension for a robust implementation.

Hardware Implementation

WALLY, the hardware demonstrator, consists of four arms with two links each (connected by revolute joints), and a square body to house the microcontroller and circuity. As shown in Figure 7, the implementation resembles the problem formulation. The end effectors consist of $24\ V$, $1\ in$ diameter electromagnets attached to the limb by a ball bearing and shaft which allows for free rotation of the magnet. Each joint is operated by a high-torque metal gear servo. There are four servos housed on the body to operate the first joint of each leg, and a servo housed on each of the first links to operate the second joint of each limb. The distance between all the joints (axles of the servos) is $4\ in$. The servos have angle range of approximately 120 degrees.

The electromagnetic end effectors are used to adhere to randomly spaced $1\ in$ diameter steel disks epoxied onto a 4'x 8' plywood wall. Using a Kinect, MATLAB's circle-finding vision capability is used to determine the exact location of disks (`imfindcircles`) as well as WALLY's position and orientation. The kinematics of resulting plans are converted to motion commands, and electromagnet and servo commands are sent via Bluetooth to an onboard microcontroller. An umbilical provides onboard power, and a safety harness is available in case of over-adventurous climbing.

`imfindcircles` searches an image for circles of a specific radius range and light or dark color using the Circle Hough Transform, ranking them from strongest to weakest circles. All of the handholds could be found by adjusting the sensitivity of this function along with the edge threshold necessary to accept an object as a circle. The main input for this functionality is an image taken by the Kinect, and the output is the center points of the discovered handholds in pixels for display, and the center points of the handholds for path planning. (On the physical wall, taping over handholds effectively removes them from $\mathbf{W}$.) The WALLY locater functionality takes in an image of WALLY on the wall from the Kinect, processes it to be in a two dimensional measurable plane, and locates fiducials on the end effectors and shoulders again using the circle finding function.

The hardware development was largely error-free, beginning early on with development of the accompanying motion planning software: testing in parallel helped to bring the system to a finalized state fluidly. Some aspects of the hardware testbed are obviously more intended for a demonstration than actual flight use: the end effectors need replacement, perhaps ice-cores [8] or micro-spines, though this is a problem not addressed in the scope of this work. Further, the planar limitation must certainly be adjusted for actual implementation. Adaptation to a multi-dimensional platform is desirable, to traverse more varied terrain. A planar system could be used by offsetting the end effectors from the body, but this would be undesirable from a torque perspective: 3D manipulator designs are desirable for their ability to approach holds from more varied positions, and for bringing their center of mass closer to their contact points.

One concern was the lack of feedback. The motion plan was executed open loop, relying on servo accuracy. This worked fairly well in practice, though slight sliding of the electromagnetic end effectors was observed. An improved global estimation system would be required in practice; the testbed's computer vision system functioned well, but end effector position tracking was not reliable enough for control. Onboard sensing would likely also be essential in a practical system, unless accurate remote surveying was performed before and during climbing. However, as a prototype and algorithm proof-of-concept of potential functionality, the testbed performed nobly.

6. CONCLUSION

The planning algorithm proposed herein has been framed as a method for efficient real-time wall-climbing, but its applicability also holds when footfalls are limited to discrete choices, including quasi-static walking. For example, traversal of flat rocky terrain with a quadruped, where only a select few footsteps (holds) are available is a potential extension. Improvements such as accounting for additional constraints including dynamics, automating primitive generation, making the local planner more robust to non-convex workspaces, and implementing in a more efficient software environment are envisioned. Some recent work has even attempted to

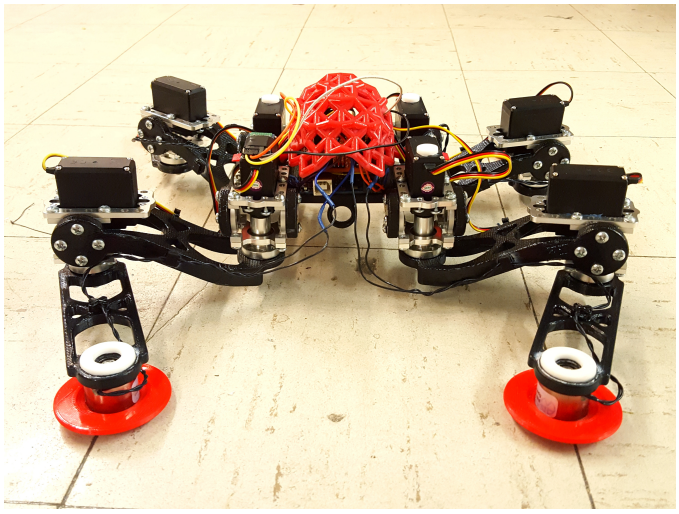

Figure 7. A front-on view of the robot hardware implementation.

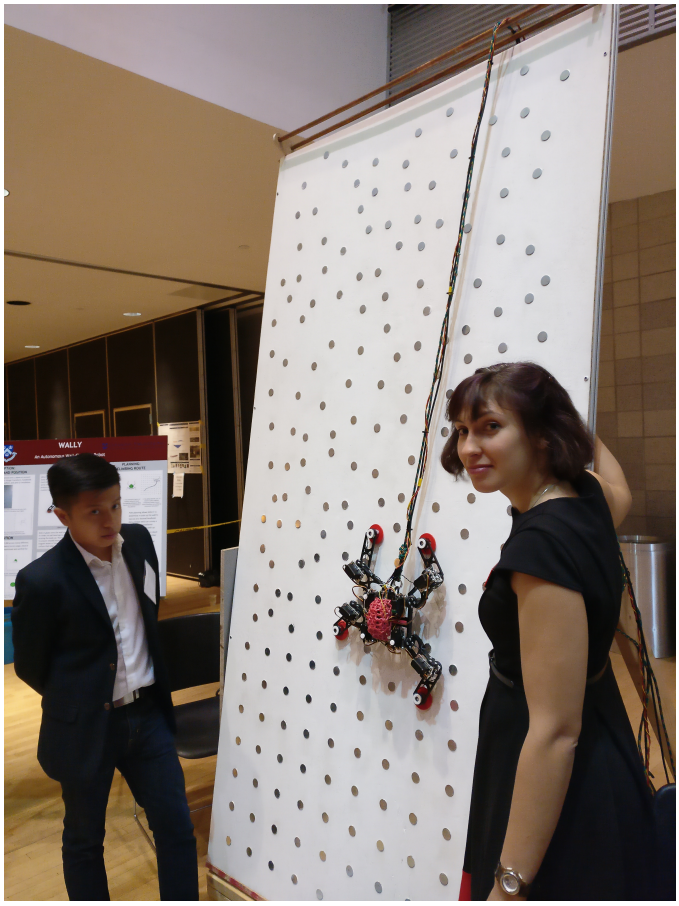

Figure 8. WALLY holding his position on a demo test wall.

apply automated motion primitive generation for humanoid robots [14], an interesting area for adaptation into a climbing motion planner.

Overall, the motion planning algorithm performed reliable, real-time quasi-static planning in environments with discrete handholds. A full-featured simulation environment has been presented, along with a proof-of-concept hardware demonstrator that successfully scaled an arbitrary vertical wall. The motion planning algorithm has clear potential for application in planetary steep terrain access robotics, and has multiple areas for potential expansion and improvement. The prospects for planetary exploration enabled by climbing robotics are exciting; reliable, efficient motion planning will by all accounts play an integral role in autonomous extreme terrain mobility of the future.

ACKNOWLEDGMENTS

The authors would like to thank Columbia Engineering for support of initial work on this project. A portion of this work was completed under NASA Space Technology Research Fellowship support, grant number 80NSSC17K0077. We would like to thank Bob Stark, Bill Miller, Mo Haroun, and Marco Nedungadi for invaluable hardware advice.

REFERENCES

[1] "NASA technology roadmaps TA 4: Robotics and autonomous systems," no. July, pp. 1–188, 2015.

[2] C. Balaguer, A. Gimenez, and A. Jordon, "Climbing robots' mobility for inspection and maintenance of 3D complex environments," *Autonomous Robots*, vol. 18, no. 2, pp. 157–169, 2005.

[3] J. Biesiadecki and M. Maimone, "The Mars Exploration Rover Surface Mobility Flight Software: Driving Ambition," *2006 IEEE Aerospace Conference*, pp. 1–15, 2006. [Online]. Available: http://ieeexplore.ieee.org/document/1655723/

[4] T. Bretl, S. Rock, and J. C. Latombe, "Motion planning for a three-limbed climbing robot in vertical natural terrain," *2003 IEEE International Conference on Robotics and Automation Cat No03CH37422*, vol. 3, pp. 2946–2953, 2003.

[5] T. Bretl, "Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem," *International Journal of Robotics Research*, vol. 25, no. 4, pp. 317–342, 2006.

[6] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 4, no. 1, pp. 1–43, 2012.

[7] B. Chu, K. Jung, C. S. Han, and D. Hong, "A survey of climbing robots: Locomotion and adhesion," *International Journal of Precision Engineering and Manufacturing*, vol. 11, no. 4, pp. 633–647, 2010.

[8] A. Curtis, M. Martone, and A. Parness, "Roving on ice: Field testing an Ice Screw End Effector and sample collection tool," *IEEE Aerospace Conference Proceedings*, vol. 2018-March, pp. 1–17, 2018.

[9] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-Time

Motion Planning for Agile Autonomous Vehicles," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 1, pp. 116–129, 2002. [Online]. Available: http://arc.aiaa.org/doi/10.2514/2.4856

[10] P. Hebert, M. Bajracharya, J. Ma, N. Hudson, A. Aydemir, J. Reid, C. Bergh, J. Borders, M. Frost, M. Hagman, J. Leichty, P. Backes, B. Kennedy, P. Karplus, B. Satzinger, K. Byl, K. Shankar, and J. Burdick, "Mobile manipulation and mobility as manipulation - Design and algorithms of RoboSimian," *Journal of Field Robotics*, vol. 32, no. 2, pp. 255–274, 2015.

[11] B. Kennedy, A. Okon, H. Aghazarian, M. Badescu, X. Bao, Y. Bar-Cohen, Z. Chang, B. E. Dabiri, M. Garrett, L. Magnone, and S. Sherrit, "Lemur IIb: A robotic system for steep terrain access," *Proceedings of the 8th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines, CLAWAR 2005*, pp. 1077–1084, 2006.

[12] S. M. LaValle, "RRT-progress and prospects," in *Algorithmic and Computational Robotics: New Directions*, 2001.

[13] S. P. Linder, E. Wei, and A. Clay, "Robotic rock climbing using computer vision and force feedback," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2005, no. April, pp. 4685–4690, 2005.

[14] K. Naderi, J. Rajamäki, and P. Hämäläinen, "Discovering and synthesizing humanoid climbing movements," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 1–11, 2017. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3072959.3073707

[15] A. Parness, N. Abcouwer, C. Fuller, N. Wiltsie, J. Nash, and B. Kennedy, "LEMUR 3: A limbed climbing robot for extreme terrain mobility in space," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5467–5473, 2017.

[16] M. Pavone and J. Starek, "Spacecraft Autonomy Challenges for Next Generation Space Missions," in *Advances in Control System Technology for Aerospace Applications*, 2014, pp. 1–34.

[17] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[18] A. Sintov, T. Avramovich, and A. Shapiro, "Design and motion planning of an autonomous climbing robot with claws," *Robotics and Autonomous Systems*, vol. 59, no. 11, pp. 1008–1019, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.robot.2011.06.003

[19] S. Tonneau, A. Del Prete, J. Pettre, C. Park, D. Manocha, and N. Mansard, "An Efficient Acyclic Contact Planner for Multiped Robots," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 586–601, 2018.

[20] R. L. Williams and B. H. Shelley, "Inverse Kinematics for Planar Parallel Manipulators," *ASME Design Technical Conferences*, pp. 1–6, 1997.

[21] M. Zucker, J. A. Bagnell, C. G. Atkeson, and J. Kuffner, "An optimization approach to rough terrain locomotion," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3589–3595, 2010.

BIOGRAPHY

Keenan Albee *is an S.M. student in the Space Systems Laboratory at MIT and a member of the SPHERES team. Keenan graduated magna cum laude from Columbia University with a BS in mechanical engineering in 2017, having conducted research in the Robotics and Rehabilitation Laboratory. Keenan's current work is in space vehicle motion planning and autonomy, under NSTRF support.*

Antonio Terán Espinoza *is a CONACyT Fellow and doctoral candidate at the MIT Space Systems Laboratory. He received his B.S. from the National Autonomous University of Mexico (UNAM), and his S.M. degree from the Massachusetts Institute of Technology as a Fulbright student. He is currently an integral member of the SPHERES group, and is working towards his Ph.D. in Aeronautics and Astronautics with a focus in autonomous navigation, mapping, and perception.*

Kristina Andreyeva *co-founded the Columbia Space Initiative and lead its inaugural NASA-sponsored mission, Micro-g NExT where she designed and machined a demonstration asteroid anchor. She graduated from Columbia University with a BS in mechanical engineering in 2017 and currently works on digital manufacturing at Voxel8.*

Nathan Werner *graduated from Columbia University with a BS in mechanical engineering in 2017.*

Howei Chen *graduated from Columbia University with a BS in mechanical engineering in 2017.*

Tamas Sarvary *is a founder at Infinite Orbits, a startup focusing on satellite servicing technologies such as GEO satellite life extension and BIU services. He graduated from Columbia University with a BS in mechanical engineering in 2017, winning the NASA Micro-g competition through the Columbia Space Initiative.*