Real-time Motion Planning in Unknown Environments for Legged Robotic Planetary Exploration

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>As Published</td>
<td><a href="http://dx.doi.org/10.1109/aero47225.2020.9172596">http://dx.doi.org/10.1109/aero47225.2020.9172596</a></td>
</tr>
<tr>
<td>Publisher</td>
<td>Institute of Electrical and Electronics Engineers (IEEE)</td>
</tr>
<tr>
<td>Version</td>
<td>Author’s final manuscript</td>
</tr>
<tr>
<td>Citable link</td>
<td><a href="https://hdl.handle.net/1721.1/129470">https://hdl.handle.net/1721.1/129470</a></td>
</tr>
<tr>
<td>Terms of Use</td>
<td>Creative Commons Attribution-Noncommercial-Share Alike</td>
</tr>
<tr>
<td>Detailed Terms</td>
<td><a href="http://creativecommons.org/licenses/by-nc-sa/4.0/">http://creativecommons.org/licenses/by-nc-sa/4.0/</a></td>
</tr>
</tbody>
</table>
Real-time Motion Planning in Unknown Environments for Legged Robotic Planetary Exploration

Keenan Albee, Alejandro Cabrales Hernandez, Oliver Jia-Richards, and Antonio Terán Espinoza
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139
{albee, cabrales, oliverjr, teran}@mit.edu

Abstract—Planetary surface robotic explorers currently implement limited amounts of autonomy, often relying on rigorously-developed offline plans. If deviation occurs, long communication delays often result in rover downtime and subsequent lost time for scientific exploration. Onboard robotic motion planning that is fast and accounts for obstacles and robot kinematics is one key piece of the autonomy pipeline required to bring more meaningful autonomy to planetary exploration. Current approaches normally rely on sampling-based planning methods like the rapidly exploring random tree (RRT) algorithm which has had considerable success for kinematic motion planning. However, global computation over the entire state space for high-dimensional systems in cluttered environments—like legged robots on a planetary surface—can be complex and too slow for practical use. What’s more, complete environment information is often not available a priori. This work proposes a real-time combined global-local planner for a legged robot in a partially unobserved, cluttered environment. Large obstacles known beforehand (e.g., orbital imagery) are accounted for using a fast global planner on a low-dimensional model. Unknown small obstacles which restrict foot placements are dealt with as they are observed using a slower but real-time local planner, obeying the complex legged robot kinematics. This approach, called SweepingRRT, makes use of observed information locally as it becomes known, while providing the fast global replanning that may be necessitated by new obstacle observations. The planning algorithm is demonstrated in simulation for a standard four-legged, eight-jointed robot in some demonstrative obstacle environments consisting of large (known) and small (unknown) obstacles using a limited sensor range.

TABLE OF CONTENTS
1. INTRODUCTION ...........................................1
2. PROBLEM FORMULATION ..............................2
3. RELATED WORK .........................................3
4. PROPOSED APPROACH .................................4
5. RESULTS .................................................5
6. CONCLUSION ............................................8
ACKNOWLEDGMENTS .......................................8
REFERENCES ..............................................9
BIOGRAPHY .............................................9

1. INTRODUCTION

Planetary surface exploration has greatly advanced the capabilities for in-situ science-gathering. While landers are limited to their original landing site and orbiters are required to observe from a distance, rovers have the ability to take measurements directly from the surface at multiple sites of interest. Currently, robotic rovers have been deployed on the Moon [1, 2] and Mars [3, 4, 5] and have greatly expanded our understanding of these celestial bodies.

All past and current planetary surface rovers have used wheels in order to provide mobility. Despite well-engineered suspension systems, wheeled mechanisms struggle with overly rough terrain which is abundant on the Moon and Mars in the form of rocks [6, 7], craters, and other steep terrain. These regions of rough terrain are sometimes also of scientific interest such as the poles of the Moon [8]. In order to access these regions, alternative forms of rover mobility need to be explored, as highlighted in the NASA Technology Roadmaps.

Legged mechanisms are a popular choice for future surface exploration robots [8, 9, 10, 11]. In contrast to wheeled mechanisms, legged mechanisms can break contact with the ground and provide high body clearance, if desired. This ability potentially allows for easier maneuvering in rough terrain but comes at the cost of increased complexity both in terms of design and operation [12]. The use of legs could also allow for the development of robots that can not only maneuver across the surface but also scale walls [11, 13]. This capability could provide access to cliffs and subterranean tunnels and dramatically increase the potential area of operations for planetary surface robotics. Further, legged robots do not suffer from the wheel degradation endured from years of rover operations over rough terrain.

Operation in areas of rough terrain requires developments in autonomous capabilities. For instance, limited visibility in cluttered environments restricts the ability of human operators to plan long trajectories [14]. The combination of limited planning ability and long communication delays results in under-utilization of rover science time and an inability to access sites of interest if they reside on rough terrain. In order to alleviate this problem, efficient motion planners for legged rovers in unknown cluttered environments must be developed. Current motion planners for wheeled rovers utilize sampling-based planning methods like the rapidly exploring random tree (RRT) algorithm [15]. However, for legged rovers the kinematic state space is much larger. In this case, computation of the full state space in a highly cluttered environment can be computationally intensive and slow. This is a particularly inhibiting limitation given that frequent global replanning might be necessary since the environment is partially observed on-the-fly.

This paper presents a combined global-local motion planning method that decouples global planning of the main robot body from the more-complicated local footstep planning. In this way, the global configuration space does not need to include states associated with the leg joints. Instead, planning of the robot’s body motion is performed using known, larger obstacles that could collide with the robot body. Footstep planning is performed locally, using the latest available sensed obsta-
Figure 1. An image of the Mars Pathfinder landing site made by the Mars Reconnaissance Orbiter (MRO) (left). Orbital imagery provides a rough idea of larger-scale geographic features and obstacles, and might be supplemented in the future by local surveyors like the Marscopter, for instance. The Pathfinder landing site is also shown in greater ground-level detail (right), where local sensors provide information about previously unknown obstacles. Here, Sojourner is shown next to the rock “Yogi,” which resides roughly in the middle of the orbital imagery [16].

The combined method allows for fast global replanning while using all available incoming obstacle information on a local planning level, with motion constrained by the legged robot kinematics.

Section 1 introduces the essential problem and the need for limbed exploration; the motion planning problem formulation is described in Section 2; Section 3 includes background knowledge and previous work related to limbed and rover motion planning from a variety of areas of robotics; the proposed algorithmic approach is presented in Section 4; simulation results on some demonstrative and more Mars-like environments are shown in Section 5; and the conclusion is presented in Section 6.

2. PROBLEM FORMULATION

There exists a robot whose configuration is fully described by the state vector, \( x \)

\[
x = [x, y, \phi, \theta]
\]

\[
\theta = [\theta_{11}, \theta_{12}, \theta_{13}, \theta_{14}, \theta_{21}, \theta_{22}, \theta_{23}, \theta_{24}]
\]

(1)

Here, \( x \) and \( y \) are the position of the center of the robot frame, \( R_F \), in the inertial frame, \( I_F \), and \( \phi \) is the orientation of \( R_F \) with respect to \( I_F \). The angles, \( \theta \), define the joint angles of the robot’s legs. The convention used here is described in Figure 2.

The robot considered may have any number of legs, but four are considered in this case since this is a commonly proposed system architecture. There exists a set of kinematic constraints, \( g_i(x, I) \), defined by the robot geometry:

\[
g_i(x, I) = 0, \ \forall i \in \{1...n\}
\]

(2)

The robot geometry parameters, \( I \), provide a convex description of the body and limbs of length \( l_{ij} \), where \( i \) is the leg index and \( j \) is the \( j \)-th component of that leg. A volumetric convex boundary around the robot is assumed, \( V_R \). In addition, a convex region around each leg’s end is defined as \( V_i \), serving as a representation of the area taken by that foot.

There exists a world, \( W \), in which the robot resides:

\[
W = \{(x, y) : w_{lx} \leq x \leq w_{rx}, w_{ly} \leq y \leq w_{ry}\}
\]

where \( w_l \) and \( w_r \) denote the bottom-left and top-right vertices in \( \mathbb{R}^2 \). Two classes of obstacles exist within \( W \): the set of all obstacles known a priori \( O_l \), and those that can be observed through the robot’s onboard perception system, \( O_s \). \( O_l \) may be provided, for example, by a satellite survey system or an orbiter spacecraft; \( O_s \) may be observed by a computer vision estimation scheme, for instance. The robot’s ability to perceive obstacles is approximated by a radius of observation, \( r_{obs} \), centered at \( (x, y) \). Obstacles that fall within range of \( r_{obs} \) are added to the observed, initially
unknown, obstacle set, $O_p$. The set of all observed obstacles, $O_{obs}$, consists of the addition of the newly observed set to the existing a priori obstacles:
\[ O_p = \{ ||r_{O_i} - r_R||_2 < r_{obs}, \forall O_i \in O_s \} \]
\[ O_{obs} = O_l \cup O_p \] (3)

The described world $W$ is shown in Figure 3.

![Figure 3. Pictorial representation of the problem’s scenario, where blue denotes the a priori known $O_l$ and observed $O_p$ obstacles, and red denotes the unknown $O_s$ obstacles; the area within the robot’s radius of observation $r_{obs}$ is also highlighted. Additionally, both the inertial $^I\mathcal{F}$ and the robot $^R\mathcal{F}$ frames are represented.](image)

The robot is given some initial state, $x_0$, and a convex goal region, $\mathcal{X}_G$, is defined. It is desired to drive the robot to some state $x_f \in \mathcal{X}_G$. In doing so, a sequence of kinematically feasible robot states must be constructed, avoiding collision with $O_l$ and avoiding foot placement on $O_p$. Optionally, it may be desirable to minimize a cost function $J(x)$ of the robot state. A description of this kinematic motion planning problem may be written:
\[
\min_x \sum_k J(x_k) \\
\text{subject to } g_{i_k}(x_k,l) = 0, \forall i \in \{1...n\}, \\
V_{i_k} \cap O_l = \emptyset, \\
V_{i_k} \cap O_p = \emptyset.
\]

Note the $k$ subscript on $g_{i_k}(x_k,l)$—the kinematic constraints for the robot change depending on whether the feet are in contact with the ground or free to move. Effectively, the robot transitions between being a parallel mechanism (when all feet are in contact) to a serial mechanism (when feet are allowed to move individually, relative to the body). This process is discussed in further detail in [13].

A pictorial representation of the planning problem is given by Figure 3. To summarize, the problem posed is that of finding a feasible (or perhaps optimal) path from a starting state $x_0$ to a goal set $\mathcal{X}_G$, subject to legged kinematic and collision constraints. The collision set $O_{obs}$ is updated in real-time.

### 3. Related Work

Multiple proposals have been brought forward for legged (and climbing) motion planning. Quadruped and bipedal footstep planning often makes use of optimization-based solvers to tackle challenging dynamic constraints, posing problems as convex programs, for instance Deits' work for the DARPA Robotics Challenge [17]. Additional works include Covariant-Hamilton approaches for improving initial trajectory guesses [18]. Other solutions discretize the problem using motion primitives [13] [19] for online execution, solving using either a graph-based or iterative solution approach. A long history of development of planetary exploration legged robots exists at NASA JPL, though these systems often rely on gaited motion or focus more on the difficult mechanical design aspects [11] [20] [18].

An RRT* approach for path planning with legged robots in rough terrain is considered in [21]. A traversability map is generated which defines the difficulty a legged robot would have in traversing a particular region. The traversability is then accounted as a cost for when determining the nearest neighbor to a new node in the RRT* algorithm. A hierarchical structure is used for the path planner. However, the difference between the global and local planner is the size of bounding box around the robot body during collision checking. No footstep planning is conducted in this approach, instead relying on the traversability map. In addition, the considered obstacles are steps and ramps which present different challenges to footstep planning than the rocky environments of planetary surfaces.

Foot placement for a legged robot traversing rough terrain is considered in [22]. Rankings for foothold selection for different types of terrain are learned offline through learning from demonstration. Footstep planning is then conducted over the next four highest-ranked footholds. While this process allowed the robot to traverse extremely rough terrain, it requires an offline step where an expert demonstrates optimal foothold selection. This can be problematic for planetary surfaces where the terrain is unknown and visibility of the types of obstacles along the path to the goal is low. In this case, the path planning algorithm might require a human in the loop which can result in significant rover downtime.

Otte and Frazzoli presented RRT-x, a replanning RRT* that accounts for newly observed obstacles while obeying robot kinematics [23]. The algorithm is asymptotically optimal and, due to its low computational cost, could be used in real-time operations. The approach considered here differs from Otte and Frazzoli by considering the foot placement when obeying the robot kinematics. Additionally, the treatment of obstacles is different, as there are distinct types of interaction with different obstacle classes (e.g., feet and $O_s$).

Many recent works have begun to consider planning in unknown environments for a variety of platforms. Tordesillas et al., for instance, address real-time safe trajectory planning with onboard observation of obstacles via an optimization approach that optimizes trajectories in the known and unknown spaces, keeping a back-up trajectory in case of imminent collision [24]. However, this approach is more tailored toward high-speed aerial vehicles and does not deal with the kinematic constraints of legged motion.

Finally, Goretkin et al. tackle planning with sensor visibility constraints to avoid entering unexplored spaces, but the approach is primarily concerned with sensor field of view limitations and safety constraints that are not readily applicable to quasi-static legged robotic motion [25].
4. PROPOSED APPROACH

There remains a need for a general-purpose motion planning method that can operate in highly cluttered environments in real-time with onboard perception, accounting for legged robot kinematics and obstacle constraints, without using predetermined footstep (handhold) positions. The approach considered in this paper uses a combined global-local planner that uses a cheap, low-dimensional system model for global planning and does out the slower, more intensive local kinematic motion planning over limited sensor horizon rollouts.

A key observation is that with a limited sensor horizon, global planning may be frequent. For instance, in the "slalom" example shown in Figure 7, the global planner frequently plans through unobserved small obstacles and must correct this information on-the-fly as small obstacles are observed. A full kinematic consideration of the robot in the global plan would be wasteful, since global plans are frequently discarded as new sensor information arrives. For example, the naive RRT approach—entailing exploration over the full 11-d state space of the robot—would result in exceptionally slow global replanning.

Rather, the complicated kinematics can be effectively separated out from the global planning. In the global planning step, the algorithm performs a path collision check of a bounding volume, $V_R$, against $O_f$ and a set of virtual obstacles, $O_v$, placed by a local planner in challenging (infeasible) regions of the terrain. Note that $V_R$ is made large enough that using a simple model of robot position in $\mathbb{R}^2$ is enough to guarantee collision feasibility of foot placements with $O_f$.

Once a global plan is formed, the local planner sweeps to update $O_f$ and plans connecting motions between the nodes of the global planner within range of $r_{obs}$. This step consists of checking that the robot kinematics (i.e., foot placements and body parallel motions) do not cause $V_i$ to collide with $O_p$. If a path is, in fact, kinematically infeasible for the full-dimensional state space then a virtual obstacle, as mentioned above, is placed and a new global plan is requested. The combined global-local algorithm is summarized in Figure 4, and a subset of it is pictorially described in Figure 5.

Global Planning

The global planning algorithm is presented in Algorithm 1, where $x \in \mathbb{X}$. The algorithm uses a modified RRT that includes collision checking. As with a typical RRT, the first step is to obtain a random point, $x_{\text{rand}}$. Nearest returns the nearest explored node, $x_{\text{nearest}}$, to $x_{\text{rand}}$. A new node is generated by the function GenerateNode by moving in the direction between $x_{\text{nearest}}$ and $x_{\text{node}}$, with a maximum preselected distance. In effect, any "fast" planning method for the low-d system would be applicable here, including mathematical optimization-based approaches. Extensions to optimizing sampling-based methods, such as FMT* and RRT* are also possible.

The next step in the RRT algorithm consists of collision checking performed by CheckCollision. The first check is that the robot maximum workspace (shown in Figure 2) centered at the newly discovered node does not collide with any large obstacles or notional obstacles. The second step is collision checking that the path connecting the nearest node, $x_{\text{nearest}}$ and the new node, $x_{\text{new}}$, does not collide with the larger obstacles or virtual obstacles. For both of these checks, $V_R$ is of maximum possible area occupied by the robot—this is equivalent to the case that the robot limits are fully extended in either direction and represents a conservative estimate, in order to simplify local planning. The algorithm terminates when the current node is inside $\mathbb{X}_G$.

Algorithm 1 The global (simple RRT) path planner.

1: procedure GLOBALRRT($x_0, \mathbb{X}_G$)
2: $\mathcal{R}_{\text{explored}} \leftarrow \{\}$
3: $k = 1$
4: for $k \leq N$ do
5: $k = k + 1$
6: $x_{\text{rand}} \leftarrow \text{RandomPoint}()$
7: $x_{\text{nearest}} \leftarrow \text{Nearest}(\mathcal{R}_{\text{explored}}, x_{\text{rand}})$
8: $x_{\text{new}} \leftarrow \text{GenerateNode}(x_{\text{nearest}}, x_{\text{rand}})$
9: if $\neg \text{CheckCollision}(x_{\text{nearest}}, x_{\text{new}})$ then
10: $\mathcal{R}_{\text{explored}} \leftarrow \mathcal{R}_{\text{explored}} \cup x_{\text{new}}$
11: if $x_{\text{new}} \in \mathbb{X}_G$ then
12: return path($x_{\text{new}}$)
13: if $x_{\text{new}} \notin \mathbb{X}_G$ then
14: return error

Local Planning

Local planning begins when the global planner has returned a feasible plan, which must now be connected kinematically. The global plan is followed in sequence until completion, unless an infeasibility is found from one of the local plans. The function LocalPlan shown in Algorithm 2 constructs lookahead local plans, over a horizon of nodes within $r_{obs}$. The arguments required are the global planning path, the current node, and the last node in sequence within range. For each node in range, the algorithm calls the ConnectNodes function to generate a kinematically connected segment in the full legged robot state space between global nodes. If there is no configuration found by ConnectNodes, the algorithm returns an empty local path and places a virtual obstacle at
the point of failure. At this point, the global planner would be called again to generate a new global plan, repeating the procedure.

The actual footstep/kinematic planner which generates the kinematically feasible steps between nodes is shown in Algorithm 3, ConnectNodes. The algorithm begins by obtaining \( r_i \) at the initial node and \( r_{goal} \) at the final node. A simple planner that creates gaits on-the-fly in the direction of the unit vector (\( \hat{e}_i \)) from \( x_i \) to \( x_{i+1} \) is implemented here. The motion of the robot is restricted in this direction, as the global planner ensures that there is a collision-free path between the two nodes connected by a straight line. An additional requirement for this implementation is that the robot can reset to its neutral stance once it reaches each global node. This assumption can easily be relaxed, allowing for additional searches about the node if the neutral stance foot positions are violated.

The largest possible travel distance, \( d_{max} \), in the direction \( \hat{e}_r \) is computed for the parallel motion (i.e., when feet are fixed and the body translates). After following \( \hat{e}_r \) for \( d_{max} \), the feet are returned to the neutral stance. A collision check against \( O_p \) is performed and, if violated, a notional obstacle is added as mentioned above and an empty path segment is returned. This gait generation repeats until the distance between the current position, \( x_{cur} \) and \( x_f \) is less than \( d_{max} \), at which point a final connection to \( x_f \) is made. The sequence of \( x_i \) is saved, and appended to the larger local horizon plan. The system may execute the entire local horizon plan, or perhaps only a single node in a receding horizon fashion if more frequent sweeping is desired.

Algorithm 3 Local footstep planner for generating kinematically feasible connections between two nodes.

```
1: procedure CONNECTNODES(\( x_i, x_{i+1} \))
2: \( S_i \leftarrow \emptyset \)
3: \( r_{cur} \leftarrow RobotConfig(x_i) \)
4: \( r_{goal} \leftarrow RobotConfig(x_{i+1}) \)
5: \( \hat{e}_r \leftarrow DirVect(x_i, x_{i+1}) \)
6: \( d_{max} \leftarrow MaxWorkspace(\hat{e}_r) \)
7: while \( |r_{cur} - r_{goal}| > d_{max} \) do
8: \( r_{cur} = r_{cur} + d_{max}\hat{e}_r \)
9: \( x_{cur} \leftarrow RobotConfig(r_{cur}) \)
10: if SmallCollisionCheck(x_{cur}) then
11: addNotionalObs(r_{cur})
12: return None
13: \( S_i \leftarrow S_i \cup x_{cur} \)
14: return \( S_i \)
```

Overall Algorithm

The combined global-local planner, dubbed SweepingRRT, is presented in Algorithm 4 and represents a pseudocode version of the framework presented in Figure 4. After initializing the algorithm with parameters, goals, obstacles, and the initial state, GlobalRRT computes a global plan. The global plan is recomputed in only two cases: the robot is unable to place a neutral stance at a node in the sensor horizon, or the local planner in between nodes fails. The result of the algorithm is a complete plan, \( S_{complete} \) which contains a kinematically feasible plan of the robot full state vector. This plan will likely have been already partially executed along the way, since online sensor observations of \( O_f \), may have induced a global replan.

5. RESULTS

SweepingRRT was run on some canonical test environments to demonstrate its features, and was demonstrated in Monte Carlo fashion on a randomized test environment for runtime evaluation. The four-legged, eight-jointed planar robot kinematics used throughout this paper were used for these test scenarios. Figure 6 shows the planner working on...
Algorithm 4 The combined global-local uncertainty-aware planner.

1: procedure SWEEPINGRRT($x_0, X_G, O_l$)
2: \[ S_{\text{complete}} \leftarrow \emptyset \]
3: \[ i_{\text{cur}} \leftarrow 0 \]
4: \[ S_{\text{global}} \leftarrow \text{GlobalRRT}(x_0, X_G) \]
5: \[ \text{while } x_{\text{cur}} \notin X_G \text{ do} \]
6: \[ O_p \leftarrow O_p \cup \text{ObserveObs}() \]
7: \[ \{i_{\text{cur}}, i_{\text{end}}\} \leftarrow \text{NodesinRange}() \]
8: \[ \text{if } \neg \text{NodesValid}() \text{ then} \]
9: \[ i_{\text{cur}} \leftarrow 0 \]
10: \[ S_{\text{global}} \leftarrow \text{GlobalRRT}(x_{\text{cur}}, X_G) \]
11: \[ \text{continue} \]
12: \[ \text{else} \]
13: \[ S_{\text{local}} \leftarrow \text{LocalPlan}(S_{\text{global}}, i_{\text{cur}}, i_{\text{end}}) \]
14: \[ \text{if } S_{\text{local}} \text{ is None then} \]
15: \[ i_{\text{cur}} \leftarrow 0 \]
16: \[ S_{\text{global}} \leftarrow \text{GlobalRRT}(x_{\text{cur}}, X_G) \]
17: \[ \text{continue} \]
18: \[ S_{\text{complete}} \leftarrow S_{\text{complete}} \cup S_{\text{local}} \]
19: \[ x_{\text{cur}} \leftarrow \text{AdvanceRobot}() \]
20: \[ i_{\text{cur}} = i_{\text{cur}} + 1 \]
21: \[ \text{return } S_{\text{complete}} \]

Figure 6. Progression of SweepingRRT for planning a path around observed obstacles representing an inconvenient pile of rocks. The global plan is shown in red, the blue circles are observed obstacles, and the green rectangle is $X_G$. An environment with only small obstacles. Since there are no large obstacles, the robot simply plans a path from the initial state to the goal with no $O_l$ collision checking. However, due to the high density of small obstacles discovered, the robot is unable to follow the global plan. Each time the local planner fails, a virtual obstacle (yellow) is added at the corresponding robot position and the global plan is re-evaluated. This begins to nudge the global plan away from the center region and eventually the robot plans a path around the obstacles and is able to reach the goal region (green).

Figure 7 shows SweepingRRT working on a slalom course. The robot starts at the top of the domain and $X_G$ is at the bottom. Solid blue squares show $O_l$ that are accounted for by the global planner. Small obstacles, $O_s$, block passages through the environment and are initially red since they are unobserved. The global planner plans a path to the goal region only accounting for $O_l$, initially. As the robot executes the plan small obstacles are observed—turning blue on the plot—and the local planner fails to find a footstep plan through the dense regions of small obstacles. This causes virtual obstacles, $O_v$, to be added to the environment, forcing the global plan to avoid the regions of small obstacles. The robot progresses through the environment, observing the small obstacles and adding virtual obstacles when the local planner fails, finally making its way to the goal after incorporating its observed obstacle information.

To estimate the performance of SweepingRRT, the planner was run on 100 randomized environments. Each environment was 25 m x 25 m in size with 10 randomly sized and distributed large obstacles and 200 small obstacles uniformly distributed across the domain. The robot starts at the top right of the domain with the goal of reaching the bottom left of the domain. Figure 8 shows an example domain and execution of SweepingRRT. Over the 100 environments, the planner took on average 4.18 seconds of computation time to plan and replan, with a standard deviation of 2.18 seconds running on a 16 GB, 8 core Intel i7-4700MQ CPU @ 2.40GHz. Note that on hardware this computation time would be spread out over the entire execution timeframe of the robot—the first local plan to begin execution is available almost instantaneously.
Figure 7. Progression of SweepingRRT for planning a path through a slalom course. Note how small unobserved obstacles (red) are converted to observed obstacles (blue) as they come within sensor range.
6. CONCLUSION

Legged robotic exploration may open the door to accessing previously unreachable terrain. Key to these systems is reliable autonomy—including motion planning—that will improve operational efficiency and safety, in particular by operating in real-time and acknowledging limitations such as incomplete environment information and limited sensor range.

This work proposes a real-time, uncertainty-aware motion planning algorithm for legged robots with onboard sensing capability. A combined global-local planner denoted SweepingRRT is introduced, in which obstacles known a priori are accounted for using a fast, low-dimensional global planner, while small, unobserved obstacles are disregarded. Instead, a localized footstep planner is used in order to validate the global plan in the presence of small obstacles as they are observed, and to pass along infeasibility information to the global planner in replanning steps. SweepingRRT is demonstrated on a variety of environment types in real-time using the legged robot kinematics.

There are a variety of improvements and future directions to pursue for this work. Namely, the local planner implemented in this paper is not robust to complex small obstacle observations, and therefore makes global replan requests more frequently than is necessary. A few ways to address this shortcoming are motion primitive-based replanning within the bounding box guaranteed by the global plan, or convex optimization-based footstep planning. The authors are currently pursuing such improvements. Further, the current global planning is non-optimizing, though this could easily be improved by using an optimizing sampling-based planner or a fast mathematical optimization-based framework. Finally, real hardware systems will likely face a number of additional constraints, such as classifying terrain types for stepping and seeking areas of greater scientific interest. These constraints offer interesting directions for future work in environment-aware and information-gain motion planning for legged robots.

ACKNOWLEDGMENTS

Funding for this work was provided by CONACyT México and the NASA Space Technology Mission Directorate through NASA Space Technology Research Fellowships under grants 80NSSC17K0077 and 80NSSC18K1186. The authors gratefully acknowledge the support that enabled this research.

REFERENCES

BIOGRAPHY

Keenan Albee is a doctoral candidate in the Space Systems Laboratory at MIT and a member of the SPHERES team. He graduated magna cum laude from Columbia University with a B.S. in mechanical engineering in 2017 and received his S.M from MIT AeroAstro in 2019. Keenan’s current work is in space vehicle motion planning and autonomy, under NSTRF support.

Alejandro Cabrales Hernandez is an S.M. candidate in the Space Systems Laboratory at MIT. He earned his S.B. from MIT AeroAstro in 2018. His current research focuses on trajectory generation for in-space robotic assembly and trajectory optimization for docking with tumbling objects. His research interests include real-time optimization and orbital dynamics.

Oliver Jia-Richards is a doctoral candidate and NASA Space Technology Research Fellow in the Space Propulsion Laboratory at MIT. He earned his S.B. and S.M. in aeronautical and astronautical engineering from MIT. His current research focuses on the use of electro-spray thrusters for guidance and control of small spacecraft during proximity operations around small asteroids.

Antonio Terán Espinoza is a CONACyT fellow and doctoral candidate at the MIT Space Systems Laboratory and Marine Robotics Group. He received his B.S. from the National Autonomous University of Mexico (UNAM), and his S.M. degree from MIT as a Fulbright student. He is currently an integral member of the SPHERES team, and is working towards his Ph.D. in AeroAstro with a focus in autonomous navigation, mapping, and perception.