IEEE Transactions on Field Robotics (T-FR) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-FR paper.
IEEE Transactions on Field Robotics
Special Issue on Space Robotics

Received 21 February 2025; revised 1 October 2025; accepted 23 November 2025.
Date of publication 2 December 2025; date of current version 10 February 2026.
This article was recommended by Associate Editor Teresa Vidal-Calleja.

Digital Object Identifier 10.1109/TFR.2025.3639455

CRESCENT: Collision-Free Highly Constrained Trajectory Optimization for Driving on the Moon

ABHISHEK CAULIGI[1], KEENAN ALBEE[2] (Member, IEEE), JEAN-PIERRE DE LA CROIX[3], AND ROLAND BROCKERS[3]

[1]Department of Mechanical Engineering, Johns Hopkins University, Baltimore, MD 21218 USA
[2]Department of Astronautical, Aerospace and Mechanical Engineering, University of Southern California, Los Angeles, CA 90089 USA
[3]Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109 USA
(Abhishek Cauligi and Keenan Albee contributed equally to this work.)
CORRESPONDING AUTHOR: KEENAN ALBEE (kalbee@usc.edu)

This work was supported by the National Aeronautics and Space Administration under Grant 80NM0018D0004.

(Regular Article)

ABSTRACT Rovers have been a mainstay of planetary exploration missions, significantly expanding our knowledge in planetary science. However, past rover missions have involved significant human supervision to oversee rover operations, a state-of-practice that scales poorly for the next generation of missions. In this work, we present the development of Constrained Roving Exploration via Safe Collision-free and Environment-aware Trajectory optimization (CRESCENT), a motion planning algorithm developed for the upcoming multiagent Cooperative Autonomous Distributed Robotic Exploration (CADRE) Lunar rover mission. CRESCENT was designed to safely drive a miniature rover platform in a highly cluttered unmapped Lunar environment, executing complex motion directives from CADRE's team-level autonomy while meeting far stricter dynamical and temporal constraints than existing onboard planetary rover planning algorithms are capable of satisfying. Our hierarchical approach formulates an efficient numerical trajectory optimization-based motion planning algorithm that makes use of nonlinear optimization to solve the planning problem in real time. We demonstrate the efficiency of our proposed approach through extensive simulations and hardware testing in a representative Lunar environment. Following CADRE's upcoming deployment on the Lunar surface, CRESCENT will be the first nonlinear optimization-based trajectory optimization approach used on another celestial body.

INDEX TERMS Model predictive control, motion planning, planning under uncertainty, space robotics, trajectory optimization.

I. INTRODUCTION

SURFACE rovers have played an extensive role in Lunar and planetary exploration to further our understanding of the solar system. Mars rover missions such as NASA's Curiosity and Perseverance have driven tens of kilometers through their mission lifetimes and have greatly improved human understanding of Martian planetary science [1], [2]. However, Mars rover operations still require significant ground-in-the-loop involvement, with ground operators involved in directing the rover across difficult terrain or particularly cluttered environments, thereby limiting the driving speeds and distances traveled by the rover and driving up operational costs. Such approaches to rover operations scale poorly for missions with multiple agents, where manual interventions for individual rovers would greatly slow down mission operations. To address these real-time operational requirements, fully onboard autonomous navigation promises to be an enabling capability for upcoming mission concepts [3].

For example, the upcoming Lunar rover mission Cooperative Autonomous Distributed Robotic Exploration (CADRE) will consist of three Lunar rovers, each equipped with a ground penetrating radar (GPR) instrument, and aims to carry out distributed mapping of the Lunar subsurface [5].

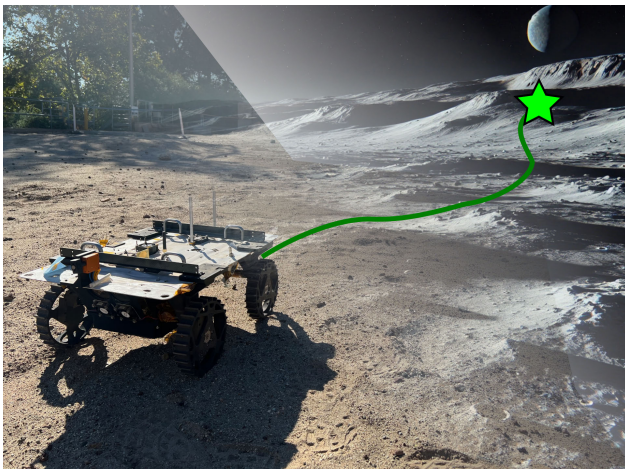


FIGURE 1. CRESCENT motion planning algorithm was developed to provide safe driving trajectories in partially unknown Lunar environments. A rover equipped with CRESCENT uses a two-stage, hierarchical planning framework that solves a trajectory optimization problem to compute trajectories and rover motion commands.

As a part of its mission operations, CADRE calls for having each rover simultaneously navigate to specified locations at precise times to effectively carry out its scientific measurements. For this mission, onboard autonomy for the rover motion planning problem is a necessity to be able to avoid collision with obstacles discovered in situ and to find trajectories that can guide the rover to the desired measurement locations with minimal ground operator intervention.

Indeed, the motion planning problem is a crucial component of future Lunar autonomy efforts that has received relatively little attention and autonomous surface mobility has been highlighted as a key area of technology development for future NASA missions [6]. For example, prior Lunar rover missions such as the Pragyan rover [7] carried out onboard state estimation, but solving the rover planning problem relied largely on ground-in-the-loop approaches, and a similar dichotomy is proposed for the upcoming Volatiles Investigating Polar Exploration Rover (VIPER) [8] mission as well. Where autonomous onboard planning has been considered for planetary rover missions, prior approaches have traditionally employed search-based approaches that use a simplified kinematic model of the rover and evaluate candidate spatial trajectories for feasibility. However, such simple kinematic planning techniques fall short for proposed missions such as the Endurance concept [9], which requires faster drive speeds exceeding $1\,\mathrm{km\,h^{-1}}$, and such missions will demand more performant real-time onboard motion planning capabilities.

For CADRE, a high-level Team Planner provides plan directives to accomplish the mission objectives and, during what is known as the formation-driving phase of the mission, these directives can include timed waypoints for each rover to follow. During this formation-driving phase, the Team Planner also provides keep-in corridors that further specify a tight spatiotemporal profile for the rover formation to allow for more accurate GPR measurements. Although existing search-based planners for planetary rover systems can plan obstacle-free paths for each rover in this formation, they fall short in being able to satisfy the tight spatiotemporal and dynamical constraints specified by the Team Planner.

As such, the motion planner necessary for CADRE and similar future autonomous exploration missions must be capable of satisfying the following desiderata.

1) *Reconciling Competing Objectives:* The planner must be capable of satisfying competing objectives, namely, adhering to higher level Team Planner directives while avoiding collisions with surrounding obstacles and other agents.
2) *Satisfying Vehicle Constraints:* The planner must be cognizant of the vehicle system and driving limits. These include both the maximum speeds the rover can drive, but also include constraints such as avoiding vehicle "dig-in" into the Lunar regolith.
3) *Handling Map Uncertainty and Clutter:* The planner will have little a priori environment knowledge and must safely and efficiently handle regions which are unknown, for which insufficient online data has been gathered, and cluttered hazardous regions.
4) *Fast Computational Rates:* The planner should be able to find solutions in real time and operate on resource-constrained compute platforms that are also running the individual autonomy stack.

A. STATEMENT OF CONTRIBUTIONS

In this work, we present Constrained Roving Exploration via Safe Collision-free and Environment-aware Trajectory optimization (CRESCENT), a trajectory optimization-based approach used as the primary onboard planner on each of the rovers in the upcoming CADRE Lunar rover mission (Fig. 1). CRESCENT, overviewed in Fig. 2, serves as a practical motion planning solution uniquely capable of meeting the aforementioned desiderata and also lays the foundation for a general-purpose planning approach for constrained planetary rover systems operating in previously unmapped environments. The key innovations of our proposed approach are given as follows.

1) An optimization-based motion planning framework consisting of a hierarchical Global and Local Planner approach that collaboratively resolves map uncertainty and enforces robustness against unexplored regions and obstacle avoidance constraints.
2) A nonlinear trajectory optimization formulation to satisfy high-level directives while enforcing a highly constrained roving problem as an efficient nonlinear least squares (NLLS) problem to enforce safety (i.e., collision avoidance) and system (i.e., rover driving limits) constraints that is solved by both the Global and Local Planners at each planning cycle.

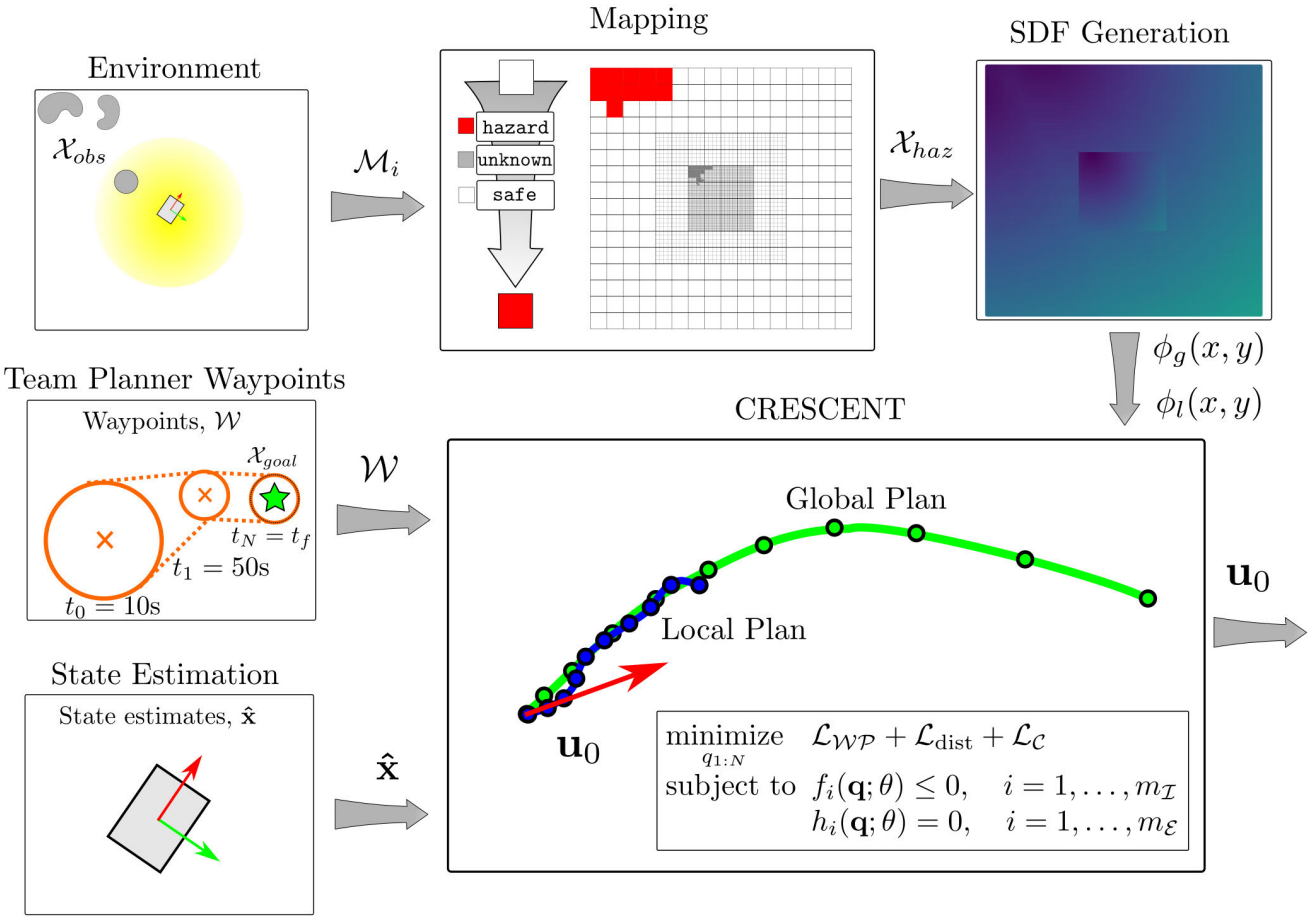


FIGURE 2. Overview of inputs to the CRESCENT algorithm. A mapping algorithm produces a multiresolution grid map from 3-D stereo vision observations, $\mathcal{M}_i$, with classifications of `safe`, `unknown`, and `hazard` [4]. Two queryable signed distance maps, $\phi_l(\cdot)$ and $\phi_g(\cdot)$, are produced based on a combination of these cell classifications, where `unknown` is treated as an obstacle for the Local Planner's SDF, $\phi_l(\cdot)$. Waypoints $\mathcal{W}$ from a high-level Team Planner and state estimates $\hat{\mathbf{x}}$ initialize the optimization problem of (15), which places nodes to form a plan that satisfies constraints. The first input, $\mathbf{u}_0$, is applied to the system.

3) The first use of signed distance functions for a flight mission and a novel factor graph formulation to statically size collision avoidance constraints used in the optimization.
4) A hierarchical framework that can operate in real time on the Snapdragon processor used on CADRE.
5) Demonstration of the efficacy of CRESCENT through exhaustive simulations and field testing in Lunar analogs at NASA Jet Propulsion Laboratory (JPL).
6) To the best of our knowledge, the first use of onboard trajectory optimization on another celestial body. In this regard, this work advances the technology readiness level (TRL) of onboard trajectory optimization as defined by NASA from TRL-4 (component testing in a lab environment) to TRL-6 (system validation in a relevant environment), prior to its Lunar demonstration.

B. ARTICLE'S ORGANIZATION

The remainder of this article is organized as follows. Section II introduces the CADRE mission and the operational constraints for which CRESCENT was developed. Section III reviews relevant literature in planning and optimal control for planetary rover systems. Section IV introduces the mathematical framework for our trajectory optimization problem. In Section V, we provide details on the hierarchical motion planning approach used in our work. Section VI presents the optimal control formulation used for planning. Section VII provides details on field tests that were conducted at NASA JPL to validate the efficacy of the proposed planner. Section VIII discusses lessons learned from the test campaign. Finally, we conclude this article and discuss future directions in Section IX.

II. CONTEXT FOR CRESCENT: MISSION OVERVIEW

CADRE is an upcoming mission to the Moon as a payload within NASA's Commercial Lunar Payload Services (CLPS) program [10] and will serve as a technology demonstration mission to showcase how enhancing multiagent surface rover autonomy enables greater scientific returns and system robustness. The mission will consist of three autonomous

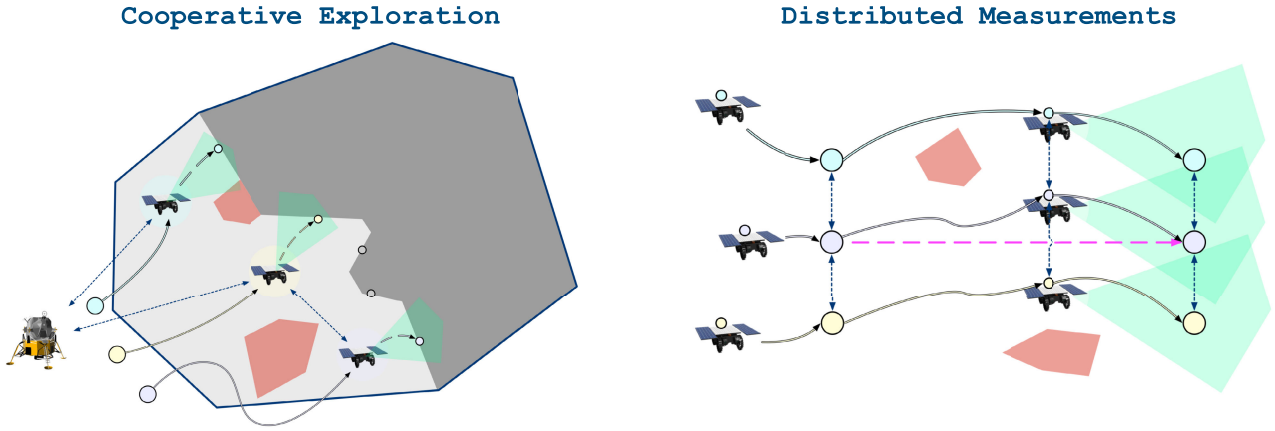


FIGURE 3. **CADRE has two distinct mission phases: cooperative exploration (exploration) and distributed measurements (formation driving). For each mission phase, the Team Planner provides waypoints for the CRESCENT motion planner on each rover to follow, shown above. Distributed measurement waypoints are far denser than exploration waypoints.**

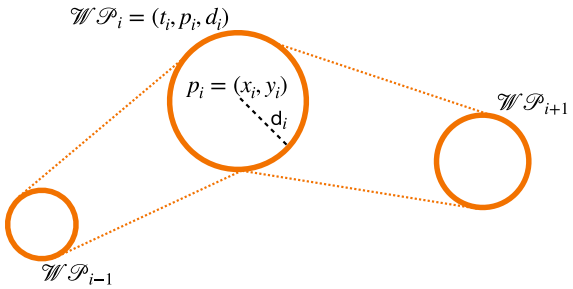


FIGURE 4. **Team Planner specifies keep-in corridor constraints for the rover to adhere to during the formation-driving phase of the mission. This keep-in corridor is constructed between two waypoints, where each waypoint is defined as a tuple $\mathcal{WP}_i = (t_i, p_i, d_i)$. The corridor is then the convex hull between two consecutive waypoints $\mathcal{WP}_{i-1}$ and $\mathcal{WP}_i$. The planner must plan trajectories that satisfy these spatiotemporal constraints for formation driving.**

Lunar rovers that are stowed on a Lunar lander. Upon deployment from this lander, the rovers will independently coordinate their operations to ensure safe and efficient mobility while collecting scientific data. Each of these Lunar rovers is equipped with a GPR instrument that allows the team of rovers to create a 3-D map of the Lunar subsurface through joined measurements. The three rovers will collaboratively investigate the region surrounding the landing site over the course of a single Lunar day, approximately 14 Earth days.

A. MISSION OVERVIEW

The mission consists of two key phases—the *exploration* phase and the *formation-driving* phase, depicted in Fig. 3 with the accompanying waypoints provided from a Team Planner for the group of agents. In the exploration phase, the rovers collectively map the surrounding area near the Lunar lander using a modified version of the multiagent motion planning algorithm presented in [11] and aggregate the local maps collected by individual rovers into a global map indicating safe, hazard, and unknown regions [12]. Next, science planners on Earth identify promising areas to carry out GPR measurements and, in the formation driving phase, a multiagent Team Planner plans waypoints for each rover to follow and collect GPR measurements. For each of these phases, each Lunar rover must be capable of planning trajectories that fulfill the mission objectives using its onboard surface rover autonomy stack. In particular, the motion planner for each individual agent needs to plan trajectories that follow the Team Planner's designated plans with safety constraints of avoiding collisions with obstacles and the other rovers.

We note that the Team Planner trajectories are quite distinct in each of these two aforementioned phases. For each agent in the exploration phase, the Team Planner plan directive consists of a single "frontier" waypoint at the boundary between known and unknown regions for a rover to travel to. There are no other spatiotemporal constraints specified, and the motion planner has an indefinite duration of time to reach this frontier waypoint. In contrast, the plan directives from the Team Planner during the formation-driving mode establish specific spatiotemporal constraints for the individual rover motion planners. In particular, the Team Planner imposes keep-in corridor constraints, as shown in Fig. 4 and defined further in Section VI-A3, to ensure interagent collision avoidance. Each keep-in corridor specifies a convex hull through which the rover must traverse. These keep-in corridors identify specific times at which the rover must pass through certain waypoints in order to ensure that accurate GPR measurements can be taken.

B. SYSTEM OVERVIEW

The rovers are designed as skid-steer vehicles and the mobility subsystem on the rover commands the rover twist, i.e., the linear velocity command v_x and angular velocity command ω_z. This 2-D twist command $u = (v_x, \omega_z)$ is then passed as individual motor commands to each rover wheel using a simple PD controller. The mobility subsystem can execute

motions with either a minimum turning radius of 30 cm or a purely rotational motion (i.e., turn-in-place). This latter turn-in-place motion is limited to a maximum of a 90° slew due to concerns about rover becoming immobilized due to "dig-in" into the surrounding Lunar regolith.

III. RELATED WORK

In recent years, autonomous surface rover planning has become a pressing need across many missions. For both the Curiosity and Perseverance rovers, AutoNav has played an outsized role in minimizing the ground-in-the-loop interventions required for navigating safely and allowing for greater science returns than otherwise possible with manual driving [2], [13], [14]. However, AutoNav uses a simple search-based planner that evaluates candidate arcs for collision and follows the subsequent path [15]. In practice, this approach often fails to find solutions in cluttered environments and stops driving until a human operator provides a path for the rover to extricate itself. Moreover, for upcoming missions such as the Endurance Lunar rover mission highlighted in the most recent Decadal Survey [16], the desire to accomplish the mission while driving at faster speeds calls for considering the full system dynamics and constraints and the use of more performant planning algorithms. Similarly, the state-of-the-art for Lunar rover driving falls far short of being able to satisfy these mission needs. As an illustrative example, the Pragyan rover tested as a part of ISRO's Chandrayaan-3 mission performs no online trajectory generation and simply follows waypoints sent by ground station operators [7]. As such, more performant autonomous trajectory generation and motion capabilities are called for the next generation of Lunar rover missions, and trajectory optimization will serve as a powerful enabling bridge between existing surface rover autonomous approaches and the capabilities required for these missions.

Trajectory optimization seeks to find *local* solutions to the optimal control problem using gradient-based numerical optimization techniques [17], [18]. Unlike sampling-based motion planning, trajectory optimization provides a richer means to encode the dynamical and spatiotemporal constraints required for CADRE. Although trajectory optimization has emerged as a cornerstone of autonomy stacks across many commercial robotic applications, it has seen limited usage in the realm of orbital and planetary robotics due to the runtime constraints on lightweight flight computers [19], [20]. Albee et al. [21] demonstrate onboard usage of nonlinear trajectory optimization for the Astrobee free-flying robot, but the computational resources available far exceed those on the more resource-constrained CADRE platform.

A key consideration in the development of these trajectory optimization approaches is in: 1) the choice of decision variables and 2) resolving the class of optimization problem to be solved at each iteration. For 1), modern trajectory optimization formulations can be distilled into one of two approaches: direct transcription-based methods that solve for both the state and control [22] or direct shooting-based methods that only solve for control [23]. Once the choice of decision variables and the optimal control formulation has been made, the class of the subsequent optimization problems, namely, whether the problem is convex or not, determines which nonlinear programming solver to use. Formulating trajectory optimization problems as a convex optimization problem has been a large area of study in the aerospace community due to the plethora of off-the-shelf solvers available to solve such problems [24], [25], [26], [27], [28]. Building upon these advances, sequential convex program (SCP) techniques have emerged in recent years as a means to develop general-purpose solvers for nonconvex trajectory optimization problems that leverage embedded convex solvers [29], [30], [31], [32]. Similarly, interfaces such as `CasADi` [33] and `acados` [34] directly provide interfaces to nonconvex solvers without the need for custom implementation of the convex solver routines

Although nonlinear optimization solvers have experienced tremendous strides, one bottleneck in deploying trajectory optimization for solving autonomy needs in aerospace applications remains compute [35]. Many of the aforementioned solvers remain too computationally expensive to deploy on the types of resource-constrained computers typically found on flight missions, though continued progress continues to produce more efficient embedded solution approaches [36], [37]. Moreover, a great deal of care and emphasis must be taken to properly populate the linear system matrix used to iteratively solve the optimization problem to take full advantage of efficient linear algebra routines [38]. To this end, factor graph-based trajectory optimization has experienced a surge of interest in recent years as a popular interface for formulating and solving trajectory optimization problems [39], [40], [41], [42]. Factor graph approaches provide a simple interface for defining the objectives and constraints for the trajectory optimization problem and efficiently solving the corresponding NLLS problem. For space applications, factor graph optimization has been extensively studied for solving localization problems [43], [44], but has yet to be used for planning and control.

IV. TECHNICAL BACKGROUND

CRESCENT solves the rover driving problem using inputs from mapping, state estimation, and higher level team planners as shown in Fig. 2. We first highlight the mathematical formulation used to state the rover driving problem as a trajectory optimization problem in continuous optimal control form, then detail CRESCENT's approach in Section V. In fixed final time form, this problem seeks to solve the following constrained optimization problem:

$$\begin{aligned} \min_{\mathbf{x}(\cdot),\mathbf{u}(\cdot)} & \int_{t_0}^{t_f} g\left(\mathbf{x}\left(t\right)\right), \mathbf{u}\left(t\right)\ dt + l\left(\mathbf{x}\left(t_f\right)\right) \\ \text{subject to } & \dot{\mathbf{x}}\left(t\right) = f\left(\mathbf{x}\left(t\right), \mathbf{u}\left(t\right)\right), \quad t \in \left[0, t_f\right] \\ & \mathbf{x}\left(t_0\right) = \mathbf{x}_{\text{init}} \end{aligned}$$

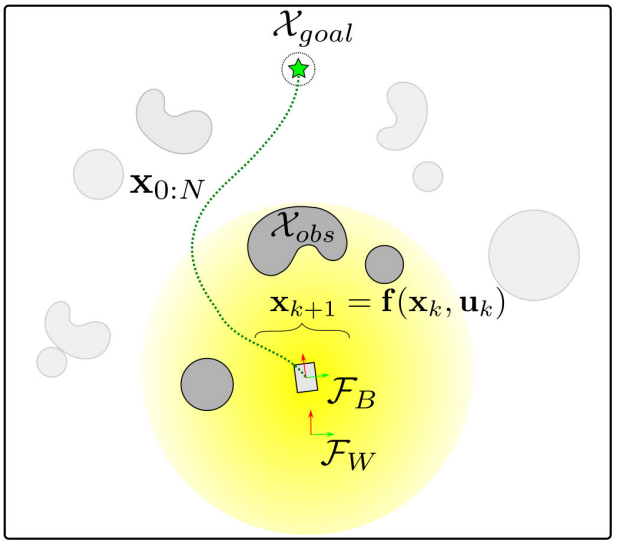


FIGURE 5. Rover driving problem is a deterministic, receding horizon trajectory optimization problem with a nonconvex, disjoint constraint set $\mathcal{X}_{obs}$. Here, $\mathcal{F}_B$ and $\mathcal{F}_W$ represent the body and world frames, respectively. The obstacle set $\mathcal{X}_{obs}$ is revealed online by a perception system, represented here as darker gray obstacles within view of the yellow perception system.

$$\begin{aligned} &\mathbf{x}\left(t_f\right) \in \mathcal{X}_N \\ &\mathbf{x}\left(t\right) \in \mathcal{X}\left(\xi\right), \quad t \in \left[0, t_f\right] \\ &\mathbf{u}\left(t\right) \in \mathcal{U}\left(\xi\right), \quad t \in \left[0, t_f\right] \end{aligned} \tag{1}$$

where $\mathbf{x} : [t_0, t_f] \to \mathbb{R}^{n_x}$ is the state trajectory, $\mathbf{u} : [t_0, t_f] \to \mathbb{R}^{n_u}$ is the control input, and $\mathcal{X}(\xi) \subseteq \mathbb{R}^{n_x}$ and $\mathcal{U}(\xi) \subseteq \mathbb{R}^{n_u}$ are constraint sets on the state and input. The parameters $\xi \in \Xi \subseteq \mathbb{R}^{n_p}$ are what vary between repeated solution attempts of the problem and are drawn from some parameter distribution Ξ corresponding to, e.g., different initial conditions and obstacle sets, among others. The running cost $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R} \to \mathbb{R}$ is associated with penalties on state and input and $l : \mathbb{R}^{n_x} \times \mathbb{R} \to \mathbb{R}$ is the terminal cost. The fixed final time t_f defines the trajectory duration. Boundary conditions are imposed upon the state trajectory at initial and final times ($\mathbf{x}(t_0)$ and $\mathbf{x}(t_f)$). In this work, we assume that the system dynamics $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R} \to \mathbb{R}^{n_x}$ are in general nonconvex.

To allow for practical solution methods via nonlinear optimization, this continuous-time optimal control problem from (1) is converted into discrete form

$$\begin{aligned} \min_{\mathbf{x}_{0:N}, \mathbf{u}_{0:N-1}} \quad & \sum_{i=0}^{N-1} g\left(\mathbf{x}_k, \mathbf{u}_k\right) + l\left(\mathbf{x}_N\right) \\ \text{subject to} \quad & \mathbf{x}_{k+1} = f\left(\mathbf{x}_k, \mathbf{u}_k\right) \quad \forall t \in \{0, 1, \ldots, N-1\} \\ & \mathbf{x}_0 \triangleq \mathbf{x}_{\text{init}} \\ & \mathbf{x}_N \in \mathcal{X}_N \\ & \mathbf{x}_k \in \mathcal{X}\left(\xi\right) \quad \forall t \in \{0, 1, \ldots, N-1\} \\ & \mathbf{u}_k \in \mathcal{U}\left(\xi\right) \quad \forall t \in \{0, 1, \ldots, N-1\} \end{aligned} \tag{2}$$

where $\mathbf{x}_k \in \mathbb{R}^{n_x}$ is the state, $\mathbf{u}_k \in \mathbb{R}^{n_u}$ is the input, and $f(\mathbf{x}_k, \mathbf{u}_k)$ represents the discretized dynamics. This is the finite-horizon formulation of the trajectory optimization problem. In a discretized form, a discrete set of N inputs will define the evolution of N state variables. There are, of course, many parameterizations of the trajectory optimization problem, including direct collocation, direct transcription, and other formats that forego gradient-based optimization altogether (e.g., zero-order sampling-based methods [45], [46]). The problem takes the form of a deterministic trajectory optimization problem of navigating with cluttered, nonconvex collision constraints. Fig. 5 summarizes the system problem formulation, where $\mathcal{X}_{\text{obs}}$ is revealed online by a perception system.

V. CRESCENT ALGORITHM

The Team Planner discussed in Section II provides a dense set of waypoints in formation-driving mode, or a single desired frontier waypoint per agent in exploration mode. Formation driving uses a high-level multiagent RRT* planner [47] for formation constraint and obstacle enforcement. However, the formation-driving planner is not real time and cannot react to changing obstacles observed during drives. Formation driving passes along these formation constraints in the form of a keep-in set between waypoints. Additionally, formation driving requires the enforcement of a time constraint attached to each waypoint to guarantee on-time arrival in formation. The role of CRESCENT is to follow these waypoints, $\mathcal{W}$ in Fig. 2 while enforcing additional constraints such as the rover kinematics and collision avoidance constraints. CRESCENT is now introduced in detail to reach the desired waypoints requested from the Team Planner; the trajectory optimization problem solved by its Global and Local Planners is then stated in Section VI.

A. UNKNOWN TERRAIN AND A TWO-TIERED PLANNER

We present an overview of the rover driving problem that is constructed at each iteration by a two-tiered Global and Local Planner, shown in Fig. 6. The Global Planner and Local Planner solve a similar trajectory optimization problem at each iteration to find collision-free trajectories that satisfy the spatiotemporal constraints driven by the GPR in formation-driving mode and minimum time requirements in exploration mode. However, the two layers of the planning stack differ on some significant aspects that necessitate a two-tiered approach. Nearly all of these differences are horizon-driven: the Local Planner operates where the rover is close-in to potentially hazardous collision constraints and consequently imposes additional conservatism.

1) *Planner Horizon:* The Global Planner is intended for slower, deliberative decisions and is in effect used primarily as a kinematically feasible path planner. The horizon for each planner is determined by the length scales of the multiresolution traversability map used on CADRE [4], where the Local Planner plans only

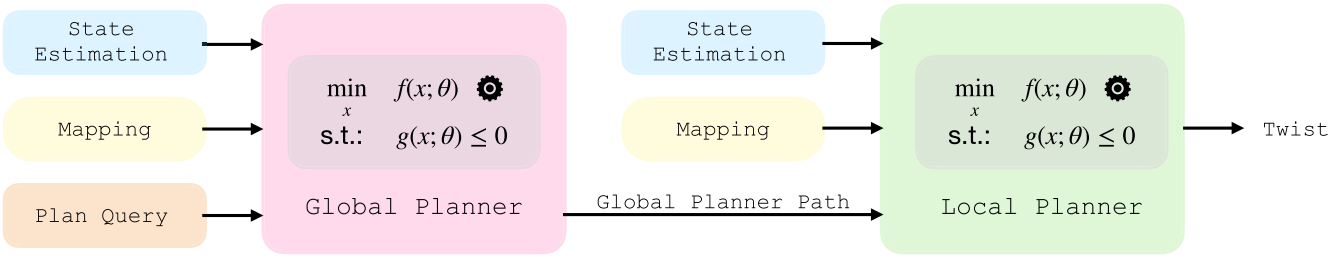


FIGURE 6. **We utilize a hierarchical, two-stage planning framework where the Global Planner computes paths that follow the Team Planner directive and the Local Planner computes the twist commands necessary to track this path. Each planner also takes as input the pose estimate computed by state estimation and an SDF computed by mapping.**

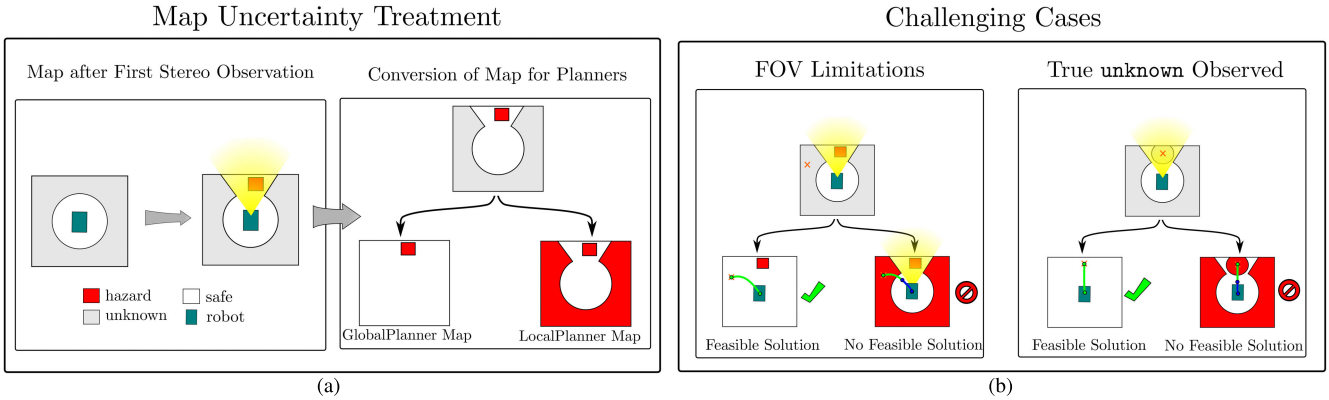


FIGURE 7. **Mapping cell classifications are interpreted differently between Global and Local Planners, resulting in two signed distance functions, $\phi_l(\cdot)$ and $\phi_g(\cdot)$. (a) Conversion of UNKNOWN map classifications into the maps used by the Local and Global Planners. (b) Inherent "optimism" of the Global Planner results in challenging cases for the Local Planner, including paths outside the FOV and unobservable cells. Example commanded waypoints and the resulting infeasible paths for the Local Planner are shown for these cases in (b).**

within the highest resolution L_0 map layer dimension (≈0.64 m) and the Global Planner plans up to the extents of the low-resolution L_3 layer (≈5.12 m). As such, the Global Planner plans approximately an order of magnitude farther than the Local Planner at each iteration.

2) *Planner Rate:* The Global Planner replans at a slower rate reflective of the mapping update rate. The Local Planner is a higher rate controller that considers quick reactivity to nearby changing obstacle constraints and dynamics uncertainty.
3) *Mapping Uncertainty:* The Global Planner treats unknown map cells "optimistically"; mapping converts these cells to free space to enable path finding, whereas the Local Planner treats these cells "pessimistically" to enable hazard avoidance for unobserved cells.
4) *Constraint Enforcement:* Some high-level constraints, such as keep-in corridors, are handled only by the Global Planner due to its longer horizon length and longer replan periods.

With maximum drive speeds of $v_{\max} = 0.05$ m/s, typical surface rover systems such as CADRE must have replan rates that are reactive to worst case constraint violation scenarios. By far, the most significant safety constraint for the rover is to avoid collision with obstacles and other agents. Consequently, the maximum distance that the rover can travel between two consecutive planner cycles can be approximated as

$$r_{\text{OL}} = v_{\max} \max\left(t_{\text{replan}}, t_{\text{update}}\right) \tag{3}$$

where t_{replan} is the planner replan period and t_{update} is the slowest planner input update rate. In our system, this is the update rate of the map $\mathcal{M}$. The r_{OL} value is most problematic in proximity to the current rover pose since decisions about obstacle avoidance cannot be made at any smaller distance unless: 1) drive speed is decreased or 2) the slower of t_{replan} and t_{update} is increased. This drives the choice of higher Local Planner replan rates and also encourages conservatism in obstacle treatment.

The most significant implication of this conservatism is the obstacle treatment detailed in Fig. 7. Point cloud estimates from the rover perception system, $R_{\text{stereo}} \in \mathbb{R}^{3\times N}$, are uncertain and, in stereo systems, degrade in certainty with distance from the cameras. The stochastic problem of mapping cell certainty is significantly simplified for the motion planner (without loss of collision conservatism) by placing map cell classifications into one of three categories {HAZARD, UNKNOWN, SAFE}. We refer the reader to [4] for the specifics of this classification process in mapping

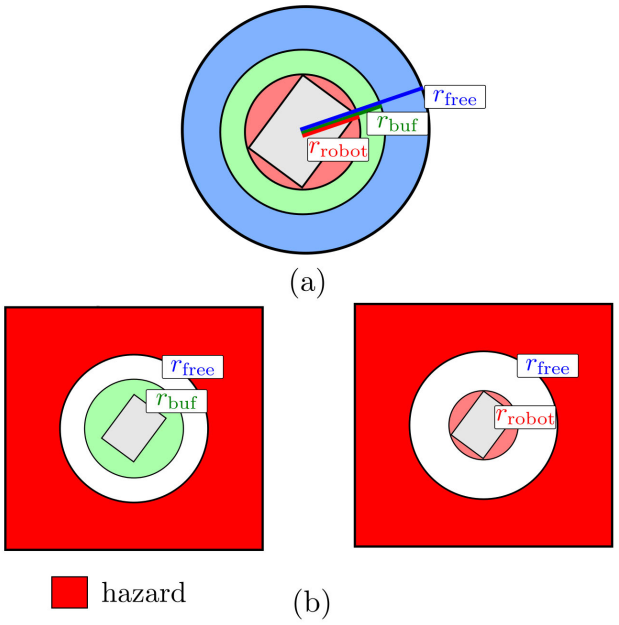


FIGURE 8. Differing rover safety radii are required within the optimizer and for final safety checks. In (a), r_{robot} inscribes the robot and r_{buf} adds a safety margin for uncertainty, while r_{free} describes the initial free area provided on the first planner call. (b) Amount of clearance space (white) that is available for the rover on initial planner startup using buffer collision checking within the optimizer (left) and minimal robot radius collision checking (right) post-optimization.

uncertain point cloud observations to a specified cell grid classification $R_{\text{stereo}} \rightarrow \mathcal{M}$. The `UNKNOWN` regions, which do not contain a sufficient number of $\mathbf{r}_{\text{stereo}}$ points for definitive classification, are treated as `SAFE` by the Global Planner, enabling discovery of new paths that might exist and preventing plan infeasibility. However, conservative collision protection is still enabled via treatment of `UNKNOWN` as `HAZARD` by the Local Planner. As detailed in Section V-A3, this model mismatch leads to challenges for the closed-loop system performance.

1) GLOBAL PLANNER

The role of the Global Planner is to interpret plan queries relayed by the Team Planner and plan paths to fulfill these planning requests. In particular, the Team Planner sends a sequence of waypoints $\{\mathcal{WP}_1, \ldots, \mathcal{WP}_N\}$ that the Global Planner takes as input to its trajectory optimization problem. Each waypoint specifies a tuple (t_i, p_i, d_i), where t_i is the time to reach the position $p_i = (x_i, y_i) \in \mathbb{R}^2$ and d_i is the ℓ_2-distance tolerance within which the rover is designated as having reached $\mathcal{WP}_i$ as shown in Fig. 4. The usage of the Global Planner markedly differs between the two mission phases described in the CADRE concept of operations as detailed in Section II. In the exploration phase, the Team Planner directive only consists of a single waypoint to drive to that is located at the frontier between known safe and unknown regions. In the Formation driving phase, the plan directive consists of a sequence of waypoints that specify the positions the rover must reach at specified intervals as well as a "keep-in" corridor within which the rover must stay to allow for accurate GPR measurements.

Rover safety radii are specified in Fig. 8, where r_{free} is an initial clearance zone applied to the map at the very first drive to provide sufficient clearance for maneuvering, r_{buf} is a buffer radius used in the collision soft constraint cost of (14), and r_{robot} is the safe robot radius used for hard constraint enforcement. Fig. 8 displays the collision radii used for cost function weighting in the optimization (left), and for hard collision checking (right) post-optimization.

Both Global Planner and Local Planner implement a persistence functionality that reduces unnecessary replan chattering. A simple cost function $J_i := \sum_{k=0}^{N-1} \ell_2(q_k, q_{k+1})$ is updated at every plan iteration i, where $J_{i+1} := J_i - l(q_{k-1}, q_k)$ reduces by the accumulated cost-to-come from the previous iteration. New plans at some future timestep j are only accepted if they offer a sufficient reduction in remaining cost-to-come, $J_j < \alpha J_i$, where $\alpha \in [0, 1]$ is the sufficient cost decrease factor. Cumulative translational path length is used as the nominal cost metric. A plan is "cached" if it offers a significant cost reduction and a cached plan may be invalidated if the system deviates significantly from the cached reference or if the cached reference is invalidated (e.g., by a new obstacle observation).

Finally, Global Planner utilizes a simple state machine consisting of four states

$$\texttt{state} := \begin{cases} \texttt{DISABLED} : \text{Stopped} \\ \texttt{IDLE} : \text{Stopped and waiting for inputs} \\ \texttt{STANDBY} : \text{Stopped and waiting at waypoint} \\ \texttt{DRIVING} : \text{Planning and sending trajectories.} \end{cases}$$

These four states were determined to allow for the Global Planner to follow the Team Planner directives. The rover enters `DRIVING` mode when actively planning toward the next waypoint and enters the `STANDBY` state to allow for the Global Planner to pause motion when the rover reaches a Team Planner waypoint $\mathcal{WP}_i$ (i.e., the rover is within the specified distance d_i, but the time t_i to reach this waypoint has not passed). The `IDLE` mode serves as the default state for the Global Planner when it is missing any of its necessary inputs (i.e., Team Planner directive, state estimate, or signed distance field (SDF) map) or when the Global Planner completes planning (either successfully or unsuccessfully). Finally, the `DISABLED` state allows for the Team Planner to temporarily disable all planning capabilities on an individual rover as necessary (e.g., in between science activities or for fault diagnosis by ground station operators).

2) LOCAL PLANNER

The Local Planner tracks the path $\mathbf{q}$ computed by the Global Planner. In Fig. 2, this Global Planner path $\mathbf{q}$ is depicted in green and provided as a reference trajectory for the Local Planner to track. We note that the Local Planner is unaware of constraints defined at the Team Planner such as keep-in

corridors and GPR-driven waypoint timing constraints. If any violations of the Team Planner keep-in corridor constraints occur, then the Global Planner sends a command to the Local Planner to immediately terminate driving. The Local Planner also receives a unique SDF map $\phi_l(x, y)$, where `UNKNOWN` map cells become marked as `HAZARD`. The output from the Local Planner to lower level controller is $u = (v_x, \omega_z) \in \mathbb{R}^2$, where v_x and ω_z are the rover translational and angular twists, respectively. These twist commands are computed by finite differencing two consecutive poses q_k and q_{k+1} along the Local Planner trajectory, where k is determined by stepping along the latest cached plan using the current time. The Local Planner state machine consists of three states: `DISABLED`, `IDLE`, and `DRIVING`. The Local Planner enters `DRIVING` mode when provided a Global Planner directive, state estimate, and SDF map and otherwise defaults to `IDLE`. The `DISABLED` mode is used by the Team Planner to fully disable Local Planner functionality.

3) CHALLENGES: RESOLVING UNKNOWN REGIONS

A key challenge in designing a performant closed-loop planner is to address occlusions and unknown regions in the map. Many rovers, and the CADRE rovers in particular, have perception systems located close to the ground and with limited field of view (FOV). Prior works have explored data-driven and perception-aware planning formulations [48], [49], but these techniques are often incompatible with real-time performance on resource-constrained embedded processors. As illustrated in Fig. 7(a), separate maps are computed for the Global and Local Planners, with the key difference being that the Global Planner SDF takes a more "optimistic" approach that classifies unknown regions as safe, as described above. This optimism is necessary for the Global Planner as the Team Planner will necessarily request driving into regions that are beyond its line-of-sight or not yet covered by stereo disparity maps. However, as indicated in Fig. 7(b), this mapping model mismatch can lead to cases that are infeasible for the Local Planner but feasible for the Global Planner. Fig. 7(b) shows this limitation for cases where an unknown region exists outside of the camera FOV (e.g., at the start of a drive, shown at left), and cases where an unobservable obstacle exists (e.g., a crater, shown at right).

This requires a perception-aware planning strategy. A simple but efficient strategy used in this work is a turning-mode behavior manager. Differential drive turn-in-place is commanded in cases where the next commanded Team Planner $\mathcal{WP}_0$ bearing angle is more than 90° offset from the current rover orientation, and in cases where any $\mathcal{WP}_i$ of the Global Planner is infeasible in the map used by Local Planner. To prevent rover "dig-in" into regolith, the turn-in-place is limited to slew a maximum of 90° and the rover must traverse at least d_{lockout} to enable `mode:= TURNING`. This approach intentionally slews the camera FOV over unobserved areas, resulting in significantly higher obstacle identification and a net reduction in `UNKNOWN` classification: the primary result is to decrease map mismatch as much as possible using the onboard perception system

$$\texttt{mode} := \begin{cases} \texttt{TURNING} : \text{Slewing to } \mathcal{WP} \\ \texttt{DRIVING} : \text{Driving using optimizer outputs.} \end{cases}$$

VI. CRESCENT: TRAJECTORY OPTIMIZATION FORMULATION

The trajectory optimization problem solved at each iteration of CRESCENT is now formally stated. Our approach entails solving a constrained, nonconvex trajectory optimization problem using an NLLS solver.

The configuration space over which we plan is $q = (x, y, \theta) \in \text{SE}(2)$. We note that because of the relatively low maximum driving speed of the rover (approximately 4 cm/s), we omit including decision variables for the control $\mathbf{u}$ directly and instead use a differential flatness-based approach where we only optimize a trajectory $\mathbf{q} = (q_1, \ldots, q_N)$. The robot kinodynamic constraints are then enforced across the configuration trajectory $\mathbf{q}$ and the corresponding velocities $\dot{\mathbf{q}} = (\dot{q}_0, \ldots, \dot{q}_{N-1})$, where $\dot{q}_k$ is computed using finite differencing.

A. OBJECTIVE FUNCTION

The objective function minimized by the trajectory optimization is comprised of three terms

$$\mathcal{L} = \mathcal{L}_{\mathcal{WP}} + \mathcal{L}_{\text{dist}} + \mathcal{L}_{\mathcal{C}}. \tag{4}$$

where $\mathcal{L}_{\mathcal{WP}}$ is a cost associated with tracking a set of reference waypoints, $\mathcal{L}_{\text{dist}}$ is a cost to encourage shortest distance driving, and $\mathcal{L}_{\mathcal{C}}$ is a cost associated with adhering to keep-in corridor constraints.

1) WAYPOINT TRACKING

Here, $\mathcal{L}_{\mathcal{WP}}$ is a regulation cost that encourages the trajectory to track the sequence of N_w reference waypoints

$$\mathcal{L}_{\mathcal{WP}} = \sum_{i=0}^{N_W} \|q_{N_i} - q_{\mathcal{WP}_i}\|_2 \tag{5}$$

where q_{N_i} is the vertex in the trajectory assigned to track the waypoint $q_{\mathcal{WP}_i}$.

2) SHORTEST PATH TRAJECTORIES

The $\mathcal{L}_{\text{dist}}$ term is a regularization term to encourage finding shortest path solutions and minimizing rover slew

$$\mathcal{L}_{\text{dist}} = \sum_{k=0}^{N-1} \|q_{k+1} - q_k\|_2. \tag{6}$$

3) KEEP-IN CORRIDORS

Finally, $\mathcal{L}_{\mathcal{C}}$ is a penalty term to induce the planner paths to lie within the centerline of the keep-in corridor specified by the Team Planner during the formation-driving phase. Each waypoint specifies a tuple (t_i, p_i, d_i), where t_i is the

time to reach the position $p_i = (x_i, y_i) \in \mathbb{R}^2$ and d_i the ℓ_2-distance tolerance within which the rover is considered as having reached $\mathcal{WP}_i$. During the formation-driving phase of the mission, the rovers are restricted to drive along a keep-in corridor to ensure that distances between the rovers lie within the necessary range for accurate GPR measurements. A keep-in corridor is defined between two consecutive waypoints $\mathcal{WP}_i$ and $\mathcal{WP}_{i+1}$ and their respective tolerances d_i and d_{i+1}. The tolerance d_i defines the ℓ_2-distance within which the rover starts from $\mathcal{WP}_i$ and the computed path must end within a distance d_{i+1} away from $\mathcal{WP}_{i+1}$. The corridor prescribed for the path to lie within is then constructed as the convex hull between these two regions specified by $(\mathcal{WP}_i, d_i)$ and $(\mathcal{WP}_{i+1}, d_{i+1})$.

Denoting this corridor between waypoints i and $i+1$ as $\mathcal{C}_{i,i+1}$, the keep-in corridor constraint is given by

$$\mathcal{L}_{\mathcal{C}} = \sum_{k=1}^{N-1} \| \mathtt{d}\left(q_k, \mathcal{C}_k\right) \|_2 \tag{7}$$

where $\mathcal{C}_k$ is the keep-in corridor associated with node q_k in the plan. We note that each node in the plan is associated with only one keep-in corridor and that the initial rover pose (i.e., q_0) is not included in $\mathcal{L}_{\mathcal{C}}$ as it is not a decision variable. Indeed, to ensure safety between agents, if the rover is detected to have violated the keep-in corridor constraint, then it immediately returns a plan failure to trigger a new plan by the Team Planner.

B. CONSTRAINTS

1) ROVER DRIVING CONSTRAINTS

For our rover platform, we enforce vehicle kinematic constraints in an implicit fashion. First, given two poses $q_k = (x_k, y_k, \theta_k)$ and $q_{k+1} = (x_{k+1}, y_{k+1}, \theta_{k+1})$, we enforce that nonholonomic constraints by using the common arc of constant curvature constraint discussed in the following [50], [51]:

$$\left\| \begin{pmatrix} \cos\theta_k + \cos\theta_{k+1} \\ \sin\theta_k + \sin\theta_{k+1} \\ 0 \end{pmatrix} \times \begin{pmatrix} x_{k+1} - x_k \\ y_{k+1} - y_k \\ 0 \end{pmatrix} \right\|_2 = 0. \tag{8}$$

We also impose turning radius constraints on the arcs driven by the rovers and denote ρ_k as the turning radius between q_k and q_{k+1},

$$\rho_k = \frac{\|d_k\|_{\ell_2}}{|\theta_{k+1} - \theta_k|} \tag{9}$$

where $d_k = (x_{k+1} - x_k, y_{k+1} - y_k) \in \mathbb{R}^2$ is the direction vector between the positions of q_k to q_{k+1}. We note that (9) introduces a singularity when $\theta_k = \theta_{k+1}$ and we avoid this issue in practice by omitting the constraint in the factor graph construction if the reference trajectory includes such a singularity (i.e., $|\bar{\theta}_{k+1} - \bar{\theta}_k| \approx 0$).

As the rovers are designed as skid-steer vehicles, the allowable turning radii that can be tracked by the rovers have to either be above some minimum value $\rho_{\min}$ or equal to exactly zero (i.e., a pure turn-in-place). This yields a disjoint constraint of $\rho \geq \rho_{\min} \cup \{\rho = 0\}$ and, while recent works have leveraged complementarity constraints [52], [53] and mixed-integer formulations [54], [55] to enforce such constraints, we relax the constraint to omit the spot turn behavior and only enforce

$$\rho_k \geq \rho_{\min}. \tag{10}$$

Instead, we encode spot turn behavior within our larger planner framework as described in Section V-A3.

We further enforce constraints for the maximum linear and angular driving speeds for the rover. The angular speed constraint is simply

$$|\theta_{k+1} - \theta_k| \leq \omega_{\max} \Delta h_k \tag{11}$$

where Δh_k is the discretization time assigned between the two nodes. For the linear driving speed, we enforce a similar constraint but also consider the driving direction to account for the nonholonomic driving constraint

$$\|d_k\|_{\ell_2} \sigma_{\text{fast}}\left(\mathtt{proj}_{\theta_k}\left(d_k\right)\right) \leq v_{\max} \Delta h_k \tag{12}$$

where $\sigma_{\text{fast}}(x) = (1 + |x|)^{-1}$ approximates the sigmoid function and $\mathtt{proj}_{\theta_k}(d_k)$ is the projection of the direction vector d_k onto the current rover bearing vector

$$\mathtt{proj}_{\theta_k}\left(d_k\right) = d_k \cdot (\cos\theta_k, \sin\theta_k). \tag{13}$$

This operation allows for the expression in (12) to project the speed onto the rover direction of travel.

2) COLLISION AVOIDANCE CONSTRAINTS

We enforce collision avoidance constraints using an SDF computed using the traversability map [4]. SDF allows for an efficient representation of the collision avoidance constraints and, to the best of our knowledge, our work represents their first use in the context of optimization-based motion planning on any flight mission. Unlike standard motion planning approaches that enumerate a discrete set of obstacles as, e.g., a composition of convex primitives [55], [56], SDF simply compute the distance to the closest obstacle as a function of the configuration space used by the planner (we refer the reader to [30] for a more thorough discussion of SDF in motion planning applications). Consequently, a point (x, y) is in collision if its signed distance value is negative, i.e., $\phi(x, y) \leq 0$, and the collision avoidance constraint is simply

$$\phi\left(q_k\right) \geq r_{\text{rover}} + \varepsilon_{\text{margin}} \tag{14}$$

where we assume a circular robot footprint with robot radius r_{rover} and $\varepsilon_{\text{margin}}$ is some nonnegative margin value.

In our work, the SDF is computed using an `OpenCV` implementation that takes the traversability map as input and computes the SDF values as a function of position only and not orientation. We note that a key advantage provided by the use of an SDF is that it allows for easily fixing the size of the trajectory optimization problem. For example, the motion planner presented in [32] with horizon N and N_{obs} obstacles enforces a total of NN_{obs} collision avoidance constraints

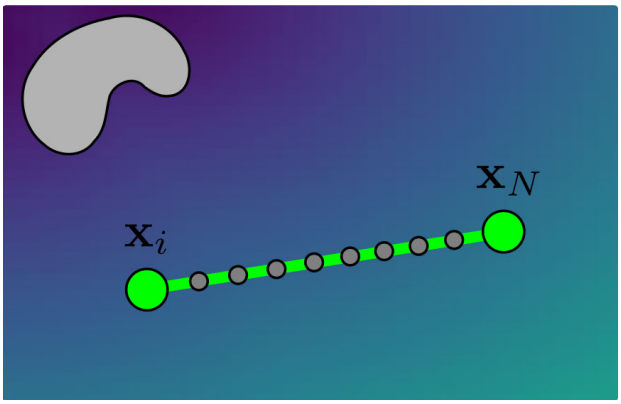


FIGURE 9. Collision checking is efficiently queried at test points between $\mathcal{WP}$. Linear interpolation is used for efficient storage and queries of gridded $\phi(x, y)$. N intermediate points $\mathbf{x}_i, \ldots, \mathbf{x}_N$ are queried between $\mathcal{WP}_i$ and $\mathcal{WP}_{i+1}$ for SDF feasibility. An example SDF is shown in the background, with color representing distance to the closest obstacle (gray).

per trajectory optimization call, a value that can change as the number of obstacles N_{obs} in the environment changes between successive planner calls. In contrast, our planner enforces N collision avoidance constraints (one per node) regardless of how cluttered the surrounding environment is and this allows us to statically size our planner. As shown in Fig. 9, the resolution of the SDF is limited to that of the traversability map and, as such, we use a simple linear interpolation scheme to evaluate the SDF at arbitrary (x, y) coordinates. We note that our choice of a circular robot footprint in computing the SDF is also determined by the design of the traversability map algorithm [4]. Collision constraints rely on a unique interpretation of mapping certainty [4] as detailed further in Section V.

C. CONSTRAINED OPTIMAL CONTROL PROBLEM

The optimal control problem solved by the Global Planner at each iteration is given by

$$\begin{aligned} &\underset{q_{1:N}}{\text{minimize}}\ \mathcal{L}_{\mathcal{WP}} + \mathcal{L}_{\text{dist}} + \mathcal{L}_{\mathcal{C}} \\ &\text{subject to Eq. (7), (8), (10), (11), (12), (14)} \end{aligned} \tag{15}$$

The solution of the Global Planner optimization problem is then passed as a reference trajectory for the Local Planner to track. The Local Planner in turn solves a smaller optimization problem corresponding to a shorter horizon to correct for deviations from the Global Planner reference trajectory and the only difference between its optimization problem is the omission of the keep-in corridor constraint (7). We note that the optimal control problem is nonconvex (e.g., kinematics constraints) and nonsmooth (e.g., C^0 SDF constraints). In practice, we find that we find high-quality feasible solutions to the optimal control problem using gradient-based optimization techniques.

D. SOLVING THE CONSTRAINED OPTIMAL CONTROL PROBLEM

To solve the constrained optimal control problem in (15), we employ an NLLS-based numerical optimization solver. Such NLLS solvers have been a mainstay of solving problems in robot localization and mapping, but have seen newfound interest in applications for robot planning and optimal control [50], [57].

We define a more generic optimization problem of the form given by

$$\begin{aligned} &\underset{\mathbf{q}}{\text{minimize}}\ f_0(\mathbf{q}; \xi) \\ &\text{subject to}\ f_i(\mathbf{q}; \xi) \le 0, \quad i = 1, \ldots, n_{\mathcal{I}} \\ &\qquad\qquad h_i(\mathbf{q}; \xi) = 0, \quad i = 1, \ldots, n_{\mathcal{E}} \end{aligned} \tag{16}$$

where there are $n_{\mathcal{I}}$ inequality constraints f_i and $n_{\mathcal{E}}$ equality constraints h_i.

To solve this constrained optimization problem using an NLLS solver, we employ a penalty method to construct a residual vector $\mathfrak{R}(\cdot)$ [58]

$$\mathfrak{R}(\mathbf{q}; \xi) = \begin{pmatrix} f_0(\mathbf{q}; \xi) \\ |f_1(\mathbf{q}; \xi)|^+ \\ \vdots \\ |f_{\mathcal{I}}(\mathbf{q}; \xi)|^+ \\ |h_1(\mathbf{q}; \xi)| \\ \vdots \\ |h_{\mathcal{E}}(\mathbf{q}; \xi)| \end{pmatrix} \tag{17}$$

where $|\cdot|^+ = \max(\cdot, 0)$. Given a reference initialization $\bar{\mathbf{q}}$, we then construct an unconstrained optimization problem to minimize the squared norm of $\mathcal{R}(\cdot)$ using Gauss–Newton type approaches to solve for local updates $\delta\mathbf{q}$

$$\underset{\mathbf{q}}{\text{minimize}}\ |\mathfrak{R}(\mathbf{q})|_2^2 \approx |\mathfrak{R}(\bar{\mathbf{q}}) + J\delta\mathbf{q}|_2^2. \tag{18}$$

After writing this term out, this subproblem solved at each iteration is given by

$$\underset{\mathbf{q}}{\text{minimize}}\ \delta\mathbf{q}^T J^T J \delta\mathbf{q} + 2\delta\mathbf{q}^T J^T \mathfrak{R}(\bar{\mathbf{q}}) \tag{19}$$

where J is the Jacobian of $\mathcal{R}(\cdot)$ with respect to $\mathbf{q}$ approximated using finite differencing. Using the necessary conditions of optimality, the stationary condition for this problem yields

$$2J^T\bar{\mathfrak{R}} + 2J^T J\delta\mathbf{q} = 0 \rightarrow \delta\mathbf{q} = -\left(J^T J\right)^{-1} J^T\bar{\mathfrak{R}} \tag{20}$$

where $\bar{\mathfrak{R}}$ is the residual vector evaluated about $\bar{\mathbf{q}}$.

This gradient update $\delta\mathbf{q}$ is applied until some convergence criterion is satisfied. In practice, to prevent runaway computation times for the optimization, users specify the number of outer loop n_{outer} iterations, the factor graph is reconstructed, and the number of inner loop iterations n_{inner} for which (20) is applied. We compare this to SCP approaches to trajectory optimization [29], [30], [31], [32] and note that n_{outer} is analogous to the number of convex approximations constructed and

Algorithm 1 CRESCENT Planner

1: **function** CRESCENT(Waypoints $\{\mathcal{WP}_1, \ldots, \mathcal{WP}_{N_W}\}$, Sufficient decrease ratio $\Delta_{\text{sufficient}}$)
2: $\bar{J} \leftarrow \infty, k \leftarrow 0, \bar{\mathbf{q}} \leftarrow \emptyset$
3: **while** `initialized` **do**
4: $x_{\text{init}} \leftarrow$ `NavigationUpdate()`
5: $\phi \leftarrow$ `MappingUpdate()`
6: **if** `InCollision`($\bar{\mathbf{q}}$) **then**
7: $\bar{J} \leftarrow \infty$
8: **end if**
9: $(\mathbf{q}^*, J^*) \leftarrow$ `TrajectoryOptimization`($\mathbf{x}_{0:N}$)
10: **if** `InCollision`($\mathbf{q}^*$) **then**
11: $J^* \leftarrow \infty$
12: **end if**
13: $\Delta \leftarrow |J^* - \bar{J}|/|\bar{J}|$
14: **if** $\Delta > \Delta_{\text{sufficient}}$ and $J^* < \infty$ **then**
15: $\bar{J} \leftarrow J^*, \bar{\mathbf{q}} \leftarrow \mathbf{q}^*$
16: **end if**
17: **end while**
18: **end function**

Algorithm 2 Factor Graph Optimization Formulation

1: **function** TRAJECTORYOPTIMIZATION(Timed waypoints $\{\mathcal{WP}_1, \ldots, \mathcal{WP}_{N_W}\}$, Signed distance field ϕ, Optimization weights $\{\omega_i\}$, Number of outer-loop iterations n_{outer}, Number of inner-loop iterations n_{inner})
2: **for** $i_{\text{iter}} = 1, \ldots, n_{\text{outer}}$ **do**
3: $\mathcal{V} \leftarrow \{\emptyset\}, \mathcal{G} \leftarrow \{\emptyset\}$
4: **for** $i_{\mathcal{WP}} = 1, \ldots, N_W$ **do**
5: **if** $i_{\text{iter}} = 1$ **then**
6: $\{x_0, \ldots, x_{n_V}\} \leftarrow \text{interp}(\mathcal{WP}_{i-1}, \mathcal{WP}_i)$
7: **end if**
8: **for** $j = 0, \ldots, n_V$ **do**
9: $\mathcal{V} \leftarrow \mathcal{V} \cup \{x_j\}$
10: **end for**
11: **if** $i_{\mathcal{WP}} = 1$ **then**
12: $\mathcal{G} \leftarrow \mathcal{G} \cup \{\mathcal{E}_{\text{bc}}(x_0)\}$
13: **end if**
14: $\mathcal{G} \leftarrow \mathcal{G} \cup \{\mathcal{E}_{\mathcal{WP}}(x_{n_V})\}$
15: **for** $j = 1, \ldots, n_V$ **do**
16: $\mathcal{G} \leftarrow \mathcal{G} \cup \{\mathcal{E}_{\text{nh}}(x_{j-1}, x_j), \mathcal{E}_{\text{tr}}(x_{j-1}, x_j)\}$
17: $\mathcal{G} \leftarrow \mathcal{G} \cup \{\mathcal{E}_{\text{v}}(x_{j-1}, x_j), \mathcal{E}_{\omega}(x_{j-1}, x_j)\}$
18: $\mathcal{G} \leftarrow \mathcal{G} \cup \{\mathcal{E}_{\text{sdf}}(x_j)\}$
19: **if** keep-in corridors active **then**
20: $\mathcal{G} \leftarrow \mathcal{G} \cup \{\mathcal{E}_{\mathcal{C}}(x_j, \phi)\}$
21: **end if**
22: **end for**
23: **end for**
24: $(\mathcal{V}^*, J^*) \leftarrow$ `g2o`($\mathcal{G}, \mathcal{V}, n_{\text{inner}}$)
25: **end for**
26: **return** $(\mathcal{V}^*, J^*)$
27: **end function**

n_{inner} the number of gradient descent steps used for solving each convex subproblem.

E. TRAJECTORY OPTIMIZATION ROUTINE

Algorithm 1 provides an overview of the trajectory optimization routine for both the Global and Local Planners. The CRESCENT planner takes as inputs the waypoints $\{\mathcal{WP}_1, \ldots, \mathcal{WP}_{N_W}\}$ defining the plan directive, the initial pose q_{init} from the state estimation pipeline, and the ϕ from the mapping pipeline (Lines 1–5). At the start of each planning cycle, CRESCENT checks whether the cached solution from the previous cycle is infeasible given the new SDF (Line 6). If the cached plan is infeasible, then this solution is cleared (Line 7). Next, the planner runs the trajectory optimization routine (Line 9) utilizing an initial guess $\mathbf{q}_0$. If this solution is collision free (Lines 10 and 11), then this new solution is accepted only if it is of sufficiently higher quality than the cached one (Lines 14 and 15).

The trajectory optimization detailed in Algorithm 2 admits as inputs the waypoints $\mathcal{WP}$ for the planner to track (i.e., the Team Planner directives are used as waypoints for the Global Planner to track and the Global Planner path defines the waypoints for the Local Planner to track). The optimization routine consists of an outer loop of n_{outer} steps (Line 2) and, for each outer loop step, a new factor graph $\mathcal{G}$ is constructed with vertices $\mathcal{V}$ that correspond to the decision variables being solved for (Line 3). For each outer loop step, the algorithm iterates through the waypoints (Line 4) and, between each pair of consecutive waypoints $\mathcal{WP}_{i-1}$ and $\mathcal{WP}_i$, creates n_V nodes (Line 6). We note that n_V is not preallocated and is determined at runtime based upon the distance between the two waypoints. Next, these nodes are added as decision variables to the graph (Line 9) and the objective function and constraints for the problem are ready to be defined. For the first node in the graph (Line 11), a boundary condition edge $\mathcal{E}_{\text{BC}}$ imposes the initial state boundary condition (Line 12). The waypoint regulation edge $\mathcal{E}_{\mathcal{WP}}$ imposes a regulation cost to induce the final node between each waypoint (x_{n_V}) to lie close to a reference waypoint (Line 14). Next, $\mathcal{E}_{\text{nh}}$ and $\mathcal{E}_{\text{tr}}$ impose the rover kinematic constraints from (8) and (10), respectively (Line 16). Edges $\mathcal{E}_{\text{v}}$ and $\mathcal{E}_{\omega}$ correspond to the linear and angular velocity contraints from (11) and (12) (Line 17). The SDF-based collision avoidance constraints are enforced via $\mathcal{E}_{\text{sdf}}$ on each node in the trajectory using the constraint (14) (Line 18). The constructed vertices $\mathcal{V}$ and factor graph $\mathcal{G}$ are passed to the `g2o` solver along with the optimization weights $\{\omega_i\}$, which carries out n_{iter} procedures of inner loop optimization steps following the update rule in (20) (Line 24).

1) INITIALIZATION

An important consideration in using trajectory optimization for solving the motion planning task is that of providing a feasible initial guess $\mathbf{q}_0$ for the nonlinear optimization. Because trajectory optimization finds local solutions to the motion planning problem, the choice of $\mathbf{q}_0$ determines the

TABLE 1. Key nominal parameters used for simulation and hardware examples.

Parameter	Value
r_{robot}	0.475 m
r_{buf}	0.075 m
$\lvert v_{\max}\rvert$	0.042 m/s
maximum planning horizon	5.12 m (GP) / 0.64 m (LP)
number of `g2o` inner loop iterations	5
number of `g2o` outer loop iterations	4

quality of the locally optimal solution found as well as the homotopy class of the trajectory. Our planner takes a simple approach to the initialization problem in order to reduce online computation: nodes are initially interpolated linearly between waypoints and are then perturbed to one side by $d_{\text{pert}} := d_0$ to move initializations away from local minima that might pass through obstacles. If a plan is infeasible, the d_{pert} initial perturbation's sign is switched and its magnitude is incremented for the next iteration, $d_{\text{pert}} = -(d_{\text{pert}} + d_{\text{inc}})$. Computationally efficient initialization was a challenge for CRESCENT, yet poor initialization strategies might lead to infeasible solutions; this approach blends ease of computation with a sufficiently rich initialization to reduce the likelihood of solutions arriving in infeasible local minima. This simple but practical solution drastically reduced the number of infeasible solutions in highly cluttered environments.

VII. EXPERIMENTS

An extensive unit test suite and accompanying closed-loop simulation environment were developed to support hardware verification. Simulation testing focused on Monte Carlo-based verifications and enumeration of performance in known challenging scenarios to reduce iteration time on hardware. Hardware testing focused on verifying performance in challenging obstacle scenarios, including craters (negative obstacles), rocks of varying sizes and distribution (positive obstacles), and mixtures of these obstacle classes with representative terrain specifications. Test campaigns were performed in the JPL Mars Yard, reconfigured for approximate Lunar terrain specification, in exploration and formation-driving modes.

A. IMPLEMENTATION DETAILS

See Table 1.

1) HARDWARE

Testing of the motion planners on hardware involved both limited testing on the rover flight models in a clean room environment and a more exhaustive test campaign using engineering (i.e., "protoflight") models. For both of these models, each Lunar rover on CADRE weighs less than 10 kg and is approximately $0.75 \times 0.5 \times 0.2$ m in volume with solar panels deployed. For sensing, the rovers are equipped with identical front and rear stereo cameras, an IMU, a sun sensor, motor odometry, and an ultrawideband ranging radio. Each rover actuates using four fixed wheels with grousers to drive in a skid-steer fashion. The engineering model of the rovers is shown in Fig. 10.

FIGURE 10. CADRE engineering model rovers driving in formation side-by-side during outdoor tests in the JPL Mars Yard. The engineering models for the CADRE mission allowed for extensive testing in outdoor environments and served as representative analogs for the flight rovers. Note wheel tracks traversed using CRESCENT.

2) SOFTWARE AND COMPUTE

Onboard computation on each rover is provided by a ModalAI VOXL, a Qualcomm Snapdragon QRB5165 development board, that features a quad-core processor with integrated GPU [5]. A custom rootfs was built and loaded with a Debian `bookworm` chroot to allow for easily supporting open-source dependencies required for the motion planner, including `g2o` [59] and `OpenCV`. The flight software algorithms are implemented in C++14 and the autonomy architecture is implemented through the use of F-Prime, an open-source framework developed at JPL that has been used in multiple small satellite missions in recent years [60], [61]. Each algorithm in the rover autonomy pipeline is implemented as a software library that is invoked by an F-Prime component. F-Prime delegates a single thread per component, which is assigned an appropriate scheduler priority and designated core on the QRB5165 based on real-time requirements. High-priority tasks are given higher priority and assigned to low-utilization cores to ensure rates are met. Task execution is driven by F-Prime's rate group system, which is tied to an FPGA-derived system clock signal. Linux CPU affinity was used to allocate all processes running on the VOXL, with defined runtime budgets allocated per process based on their accompanying F-Prime FPGA-driven rate group. The Global and Local Planners are deployed as separate threads on a single high-performance core of the QRB5165 to ensure task execution within their assigned computation cycles. The Global and Local Planners shared this core with a factor graph-based localization component and a mapping component [4].

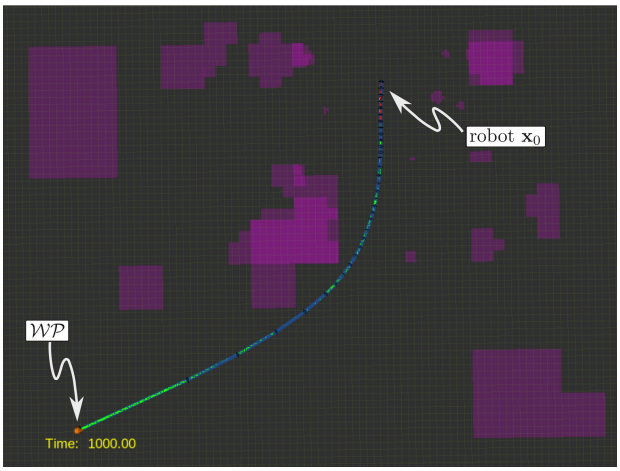


FIGURE 11. Representative example of unit testing conducted in the closed-loop motion planner simulation, depicting a single waypoint plan. Obstacles were painted into the different layers of the traversability map (purple), with waypoints (orange) and timing tested under randomized and adversarial conditions.

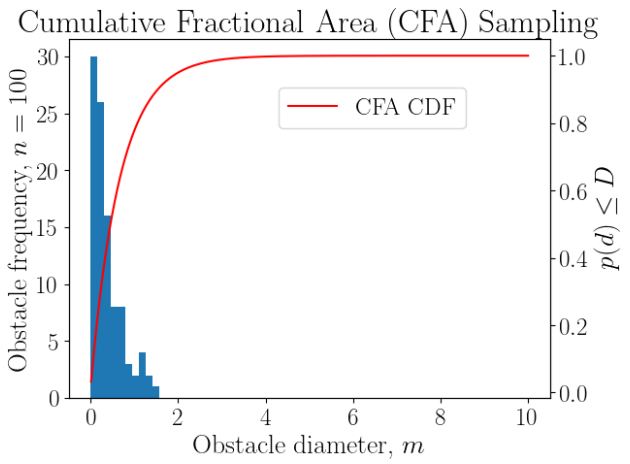


FIGURE 12. Sample draw of 100 obstacles (blue) from a CFA distribution over obstacle diameter (red) used in Monte Carlo examples.

The Global and Local Planners were allotted a runtime deadline of 1 and 0.2 Hz, respectively. Global planning rates, which account for longer horizon behavior, were selected based on maximum mapping rates, also set at 1 Hz, to avoid recomputation without additional environment updates. Similarly, Local Planner rates, which account for both local map updates *and* updated state estimates, were set to meet 5-Hz incoming state estimates to provide the fastest possible reaction to state error.

Finally, it should be noted that thread prioritization was critical to real-time operation. Since the VOXL relies on the Linux scheduler, the entire system is soft real time; rates are best effort and not guaranteed. For the most safety-critical planning component, the high-rate Local Planner was given top thread priority (-20) over the Global Planner. Both full stack testing and deployment used these priority rates to ensure that probabilistic real-time performance was achieved.

3) SOLVER DETAILS

For the `g2o` solver, we found that setting the number of outer loop iterations n_{outer} to 4 and the number of inner loop iterations n_{inner} to 5 traded off runtime and sufficient cost decrease. Edge weights $\{\omega_i\}$ used in the optimal control problem were tuned through a combination of simulation and hardware tests.

B. SIMULATION EXPERIMENTS

Simulations for planner development and testing were conducted in a custom kinematic simulator, visualized in Fig. 11. The simulation utilized `ROS`-based wrappers around our software components. This simulation environment served as a "simple sim" to test the core capabilities of the two planners, with ability to simulate varied mapping environments and the dynamics described in (8). State estimation is not simulated in this environment; instead, we provide ground truth pose values to the planner and populate the multiresolution traversability map directly (as shown in Fig. 11 in the purple colored cells). This simulation environment was used to tune the planner weights, evaluate issues discovered during hardware tests, and define benchmark unit tests. Although the full list of parameters, including tuned optimization weights, numbered more than 100 parameters, the key nominal parameters used in simulation and on hardware are shared in Table 1.

1) MONTE CARLO ENVIRONMENT SWEEP

Utilizing the simulation environment, Monte Carlo sweeps over anticipated obstacle distributions were performed to evaluate planner robustness, using a representative obstacle distribution of the Lunar terrain. Cumulative fractional area (CFA) sampling—used in the Mars exploration program [62] for accurately describing the size distribution of rocks observed over a given area providing—was selected as the distribution strategy.

The CFA models were specified as

$$N(D) = Le^{-sD} \tag{21}$$

$$F_k(D) = ke^{-q(k)D} \tag{22}$$

$$q(k) = \left(A + \frac{B}{k}\right) \tag{23}$$

where $F_k(D)$ is the CFA covered by rocks of a given diameter or larger, $N(D)$ is the cumulative number of rocks per square meter with diameter greater than D meters, L is the total number of rocks of all sizes per meter squared, and k represents the fraction of surface area covered by rocks of all sizes. A and B are fit parameters.

A cumulative distribution function over rock diameter can be derived from (23),

$$F(d) = 1.0 - \frac{F_k(D)}{k} \tag{24}$$

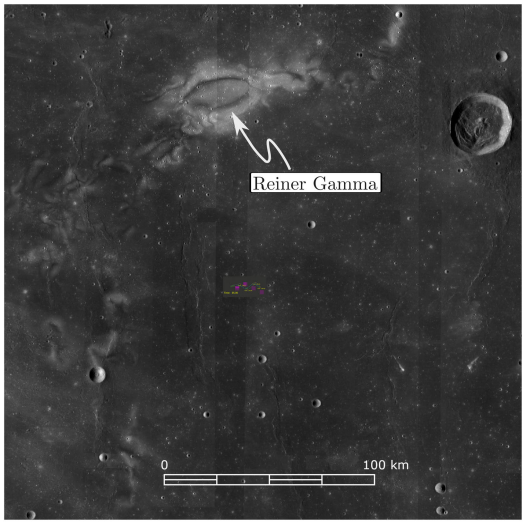


FIGURE 13. Lunar Reconnaissance Orbiter (LRO) LROC imagery of Reiner Gamma, the highest resolution imagery available of the future landing site for CADRE. Orbital observations from LRO's LROC and LOLA provide limited resolution. Credit: LRO WAC Science Team.

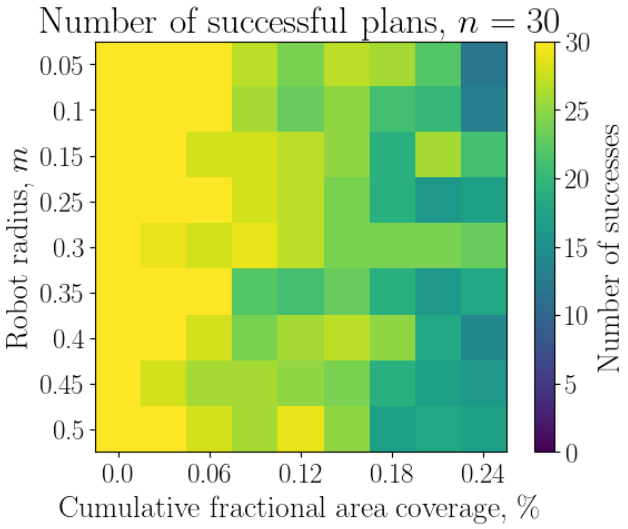


FIGURE 14. Monte Carlo testing results in the simulation environment, sweeping over rover radius *r* and CFA coverage of simulated environment realizations.

$$F^{-1}(p) = \frac{\ln(1-p)}{-q} \tag{25}$$

shown in Fig. 12 with a sample drawn from the distribution using inverse transform sampling, see (25), with $p \in [0, 1]$ for 100 samples.

Approximate CFA fit parameters $k = 0.1, A = 1.79$, and $B = 0.152$ were determined for the anticipated environment at Reiner Gamma, where the CADRE mission will operate. Fig. 13 shows the highest resolution camera imagery available of the potential landing site for the CADRE mission; environmental details are extrapolated from other in situ Lunar observations.

Sweeping over the CFA range [0, 0.24] and potential rover radii $r_{\text{robot}} \in [0.05, 0.5]$ m, Fig. 14 shows Monte Carlo

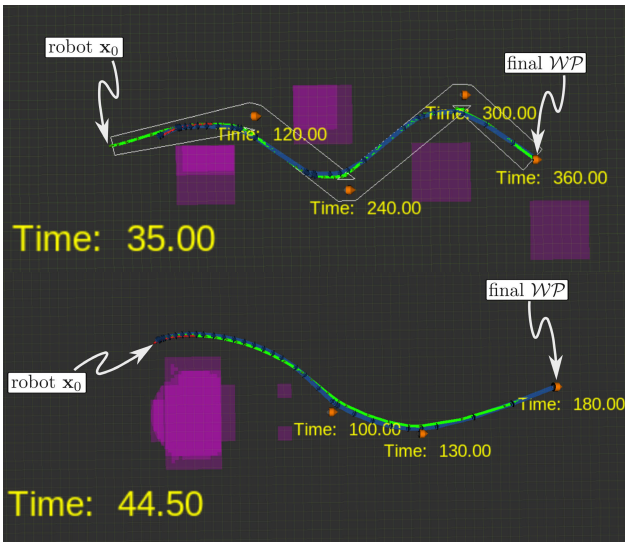


FIGURE 15. Two representative test cases: "slalom" formation-driving (top) and exploration with multiple local minima (bottom) used for cost analysis. Waypoints are shown in orange.

results for the number of instances to successfully reach the randomized goal over $n = 30$ environment samplings per parameter combination. The randomized environment measures 10×10 m.

Key parameters of this analysis are shared in Table 1. Each run is deemed successful if the rover, executing CRESCENT, reaches the goal with an environment generated from the CFA distribution. Environments are cleared of obstacles at the start and goal locations to ensure the feasibility of the planning problem. Each Monte Carlo evaluation runs a complete loop of the planning algorithm with online replanning using the latest available model knowledge (i.e., updated map information within sensor range). It is assumed that map information within range is instantly incorporated and accurately observed (i.e., occlusions and other complicated factors are not modeled). Runs are terminated after a maximum number of unsuccessful replans.

These evaluations indicate the planner's ability to satisfactorily reach the goal location in feasible environments of increasing complexity. For the actual rover safety radius of approximately 0.475 m in a worst case environment of CFA = 0.24, we have at least 21 successful plans for every 30 requested exploration waypoints; this includes environments that might be kinematically infeasible for a rover of the desired radius.

Monte Carlo results indicate a slightly diminished success rate with increasing robot radius (an indicator of increasing problem difficulty) and a mild dropoff in success rate around a CFA of 0.18. As expected, larger rover safety radii and denser environments cause more frequent planning failures. We note that terrains with higher CFA coverage are often infeasible by construction, with no collision-free trajectories for rover safety radii of any reasonable size. These results

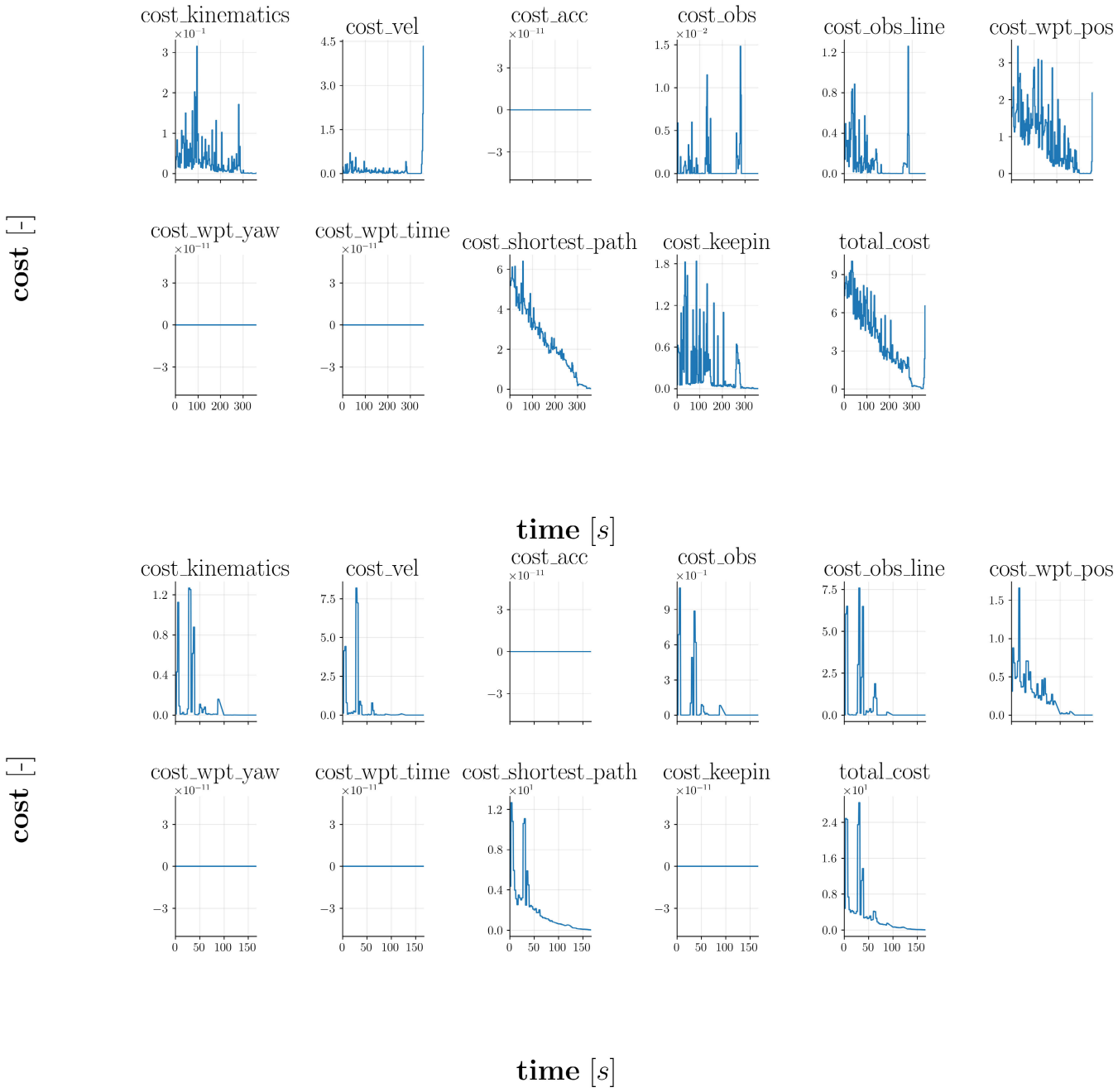


FIGURE 16. **Cost breakdown and convergence diagrams for a formation-driving (top) and exploration task (bottom). Costs were tuned using a cost sweep analysis to provide optimal total cost decrease and constraint enforcement.**

show the promise of the approach with online-updated maps in moderately cluttered terrains, even with substantial robot safety radii, lending confidence to successful execution in expected terrain specifications at Reiner Gamma.

2) COST ANALYSIS

A parameter sweep study for balancing cost weights was performed to demonstrate the weight tuning procedure required for the cost terms in exploration and formation-driving modes. Representative, challenging environments were constructed for each of these modes, including an exploration case with obstacle avoidance and multiple local minima, and a "slalom" formation-driving example requiring tight maneuvering within a cluttered environment with small timed keep-in corridors (see Fig. 15). The representative environments and their cost terms at each global planner replan are shown, broken down by every term included in the cost function. Total cost decreases, as expected, closer to the goal, which is dominated by path-minimization cost but with frequent competing cost terms activating as the system nears

FIGURE 17. Planner was evaluated over case study and representative terrains in the JPL Mars Yard on engineering model hardware of CADRE flight units. Terrains included narrow and cluttered environments (left), craters (center), and approximate terrain specification environments (right).

various constraint violations (see Fig. 16). For the exploration case, keep-in constraints are inactive and so have no contribution; for the formation-driving case, keep-ins and obstacles are active throughout to encourage safe, feasible plans. Over a dozen cost weightings were balanced to reflect optimal system total cost decrease and constraint enforcement.

C. HARDWARE EXPERIMENTS

Although testing of the surface autonomy software was conducted on the flight rovers, these tests were conducted in a clean room environment and were limited in scope due to safety precautions in handling the flight models. The bulk of our hardware testing campaign was with engineering models with identical computational, sensing, and actuation capabilities as the flight models and was carried out in the JPL Mini-Mars Yard and Mars Yard. As shown in Fig. 17, this real-world testing allowed for validating our simulation results across various configurations of Lunar obstacle environments meant to mimic the rock and crater obstacle distributions expected on flight.

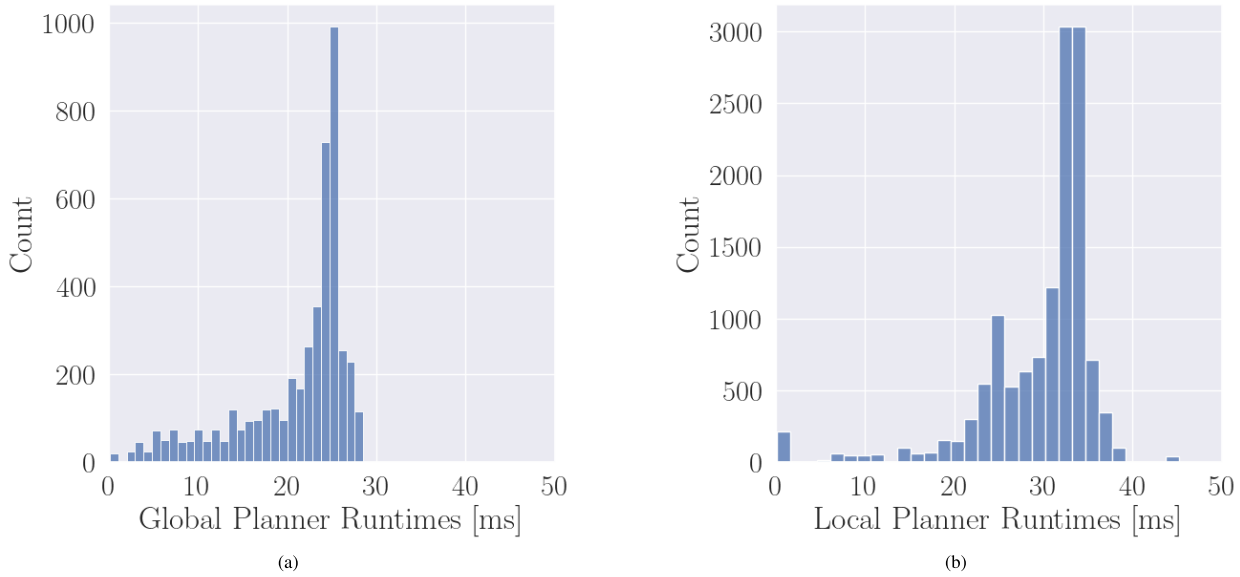
1) COMPUTATION TIME RESULTS

A key design consideration in formulating the trajectory optimization problem was to tune the optimal control problem to be able to satisfy the prescribed runtime limits. As the Global Planner and Local Planner share a core of the Snapdragon with other autonomy algorithms, excessive optimization runtimes could lead to cycle slips for the remaining FPrime components and potentially cause degraded system performance. Consequently, the Global Planner and Local Planner were prescribed upper bounds for their runtimes of 150 and 40 ms, respectively. Fig. 18 shows a histogram of planner runtimes from representative hardware experiments; the planner runtimes sufficiently satisfy the required run rates. We note for the Local Planner a set of runtime values below 10 ms. These data points correspond to planner cycles when the Local Planner commands a turn-in-place as detailed in Section V-A3 and does not trigger a `g2o` optimization call.

We also note that one advantage conferred by the use of an SDF to represent obstacles is that the complexity of the planning problem is not reflected in the optimization runtime. Namely, planning algorithms that use a clustering approach to represent obstacles add a collision avoidance for each obstacle at every time step of the trajectory optimization problem [32], [63] and this correspondingly results in NN_{obs} collision constraints for a planning problem with a horizon N. As shown in Fig. 18, the use of an SDF separates the planning complexity by only adding the single constraint from (14) at each time step for a total of N collision avoidance constraints.

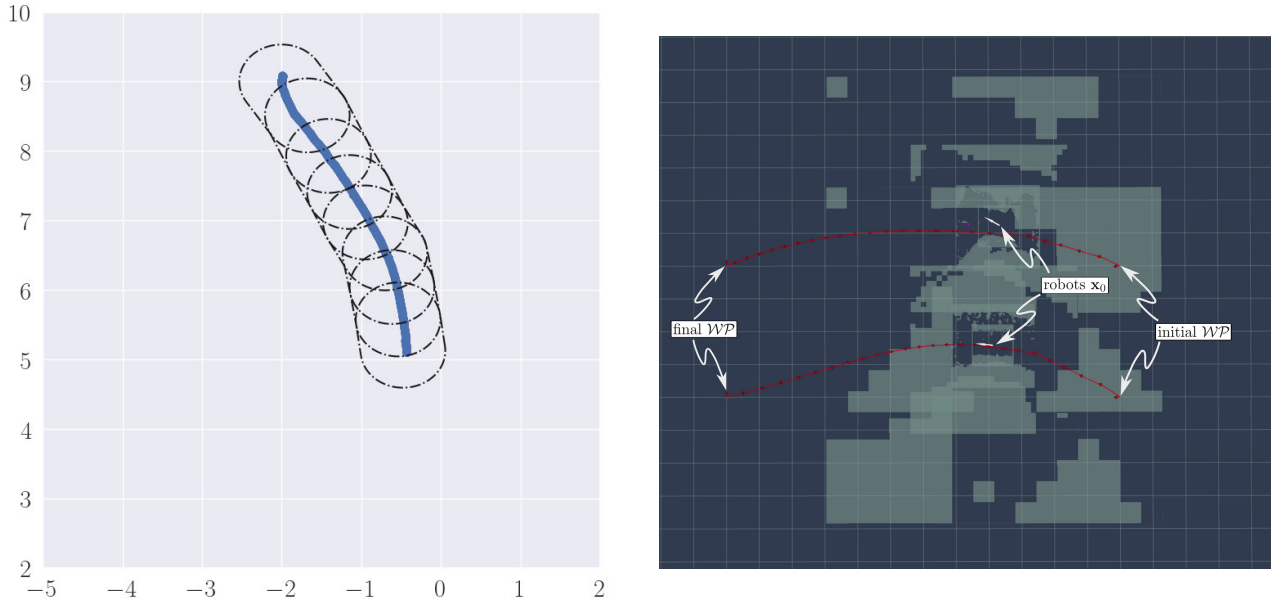
2) CASE STUDY: FORMATION DRIVING

A driving use case for the motion planner on CADRE is to satisfy the spatiotemporal constraints specified during the formation-driving mode of the mission. In this phase, the Team Planner specifies a series of position waypoints for the planner to reach at specified times and a series of keep-in corridors to lie in (i.e., the convex hull between two consecutive waypoints). To this end, we conducted extensive testing to test the performance of the motion planner during formation-driving mode. Fig. 19 shows a 4.8-m drive with an active keep-in corridor for the planner and illustrates the keep-in corridors and position waypoints shown



(a) (b)

FIGURE 18. Histogram of the runtimes logged by each motion planner. We note here that Global Planner and Local Planner runtimes are comparable, but the trajectory discretization for both is markedly different. Namely, the number of decision variables for each planner is approximately similar, but the planning horizon for the Global Planner is approximately 5.12-m, whereas the Local Planner plans trajectories over a 64-cm horizon. (a) Runtimes for Global Planner. (b) Runtimes for Local Planner.



(a) (b)

FIGURE 19. Multiagent formation-driving experiments were conducted in the JPL Mars Yard to test the efficacy of the planner in following the Team Planner generated reference trajectories. The Team Planner uses a predefined map to generate nominally collision-free trajectories, but the onboard motion planners react to hazard information updated by real-time mapping and are free to replan safe paths accordingly. (a) Reference waypoints and keep-in corridors provided by the Team Planner are shown in the dashed black line with the ensuing robot trajectory in blue. (b) Two-agent formation driving test conducted in the JPL Mars Yard with the Team Planner reference trajectories shown in red. Gray is a live record of unknown terrain across different map layers, which is fused to construct $\phi_l(.)$.

with the dashed black line and the trajectory of the rover in blue. We note that only one keep-in corridor constraint is enforced for each node in the trajectory optimization problem.

We note that in our implementation, the Local Planner is not cognizant of the keep-in corridor constraints. As such, we found that scenarios in which map occlusions triggered a turn-in-place response by the Local Planner would lead to

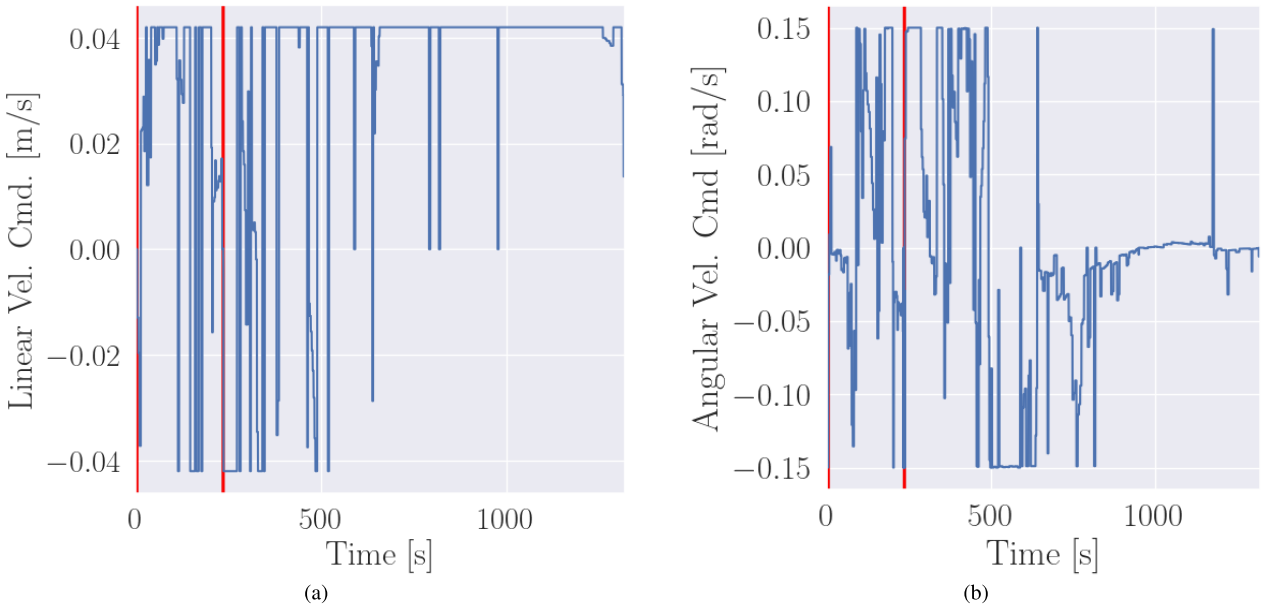


FIGURE 20. **Twist command profile for the Local Planner over a drive. Turn-in-place motions are labeled with vertical red lines where the rover issues a zero linear twist command. (a) Linear velocity commands. (b) Angular velocity commands.**

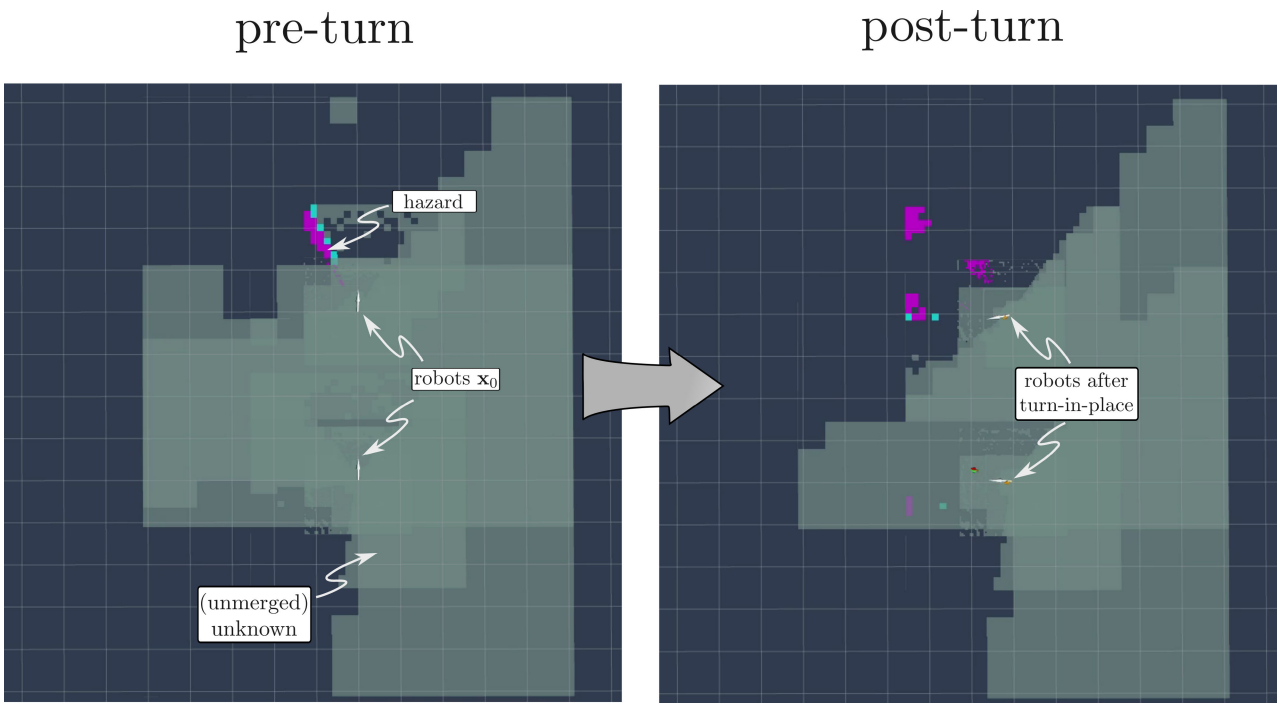


FIGURE 21. **Illustrative example of the turn-in-place behavior demonstrated during a multiagent exploration phase hardware test. Initial waypoints for each rover are off to the left (not pictured) outside the initial FOV. Both rovers execute an open-loop spot turn to orient their bearing with the waypoint heading vector and end facing left. Blue and purple squares indicate different observed hazard classifications.**

longer driving times and a greater likelihood of failure to satisfy the timing constraint of reaching position waypoints. Namely, the Local Planner triggered spot turns to resolve unknown regions of the map, but this behavior occasionally resulted in the Global Planner timing out.

3) CASE STUDY: SPOT TURN BEHAVIORS

As discussed in Section V-A3, a key challenge in integrating the motion planning pipeline with the mapping component was in how to handle unknown regions of the map. In particular, the placement of the stereo cameras on the CADRE rover

and the implementation of the mapping algorithm lead to a limited FOV in which obstacle classification is carried out. Upon start-up and after idle periods for the rover, we observed that large portions of the map would be marked as UNKNOWN and lead to Local Planner failures.

To address this, we implemented a spot turn behavior that executed a pure turn-in-place command when the Local Planner was commanded to drive into unknown regions. Figs. 20 and 21 demonstrate this behavior during a multiagent exploration test. As seen in Fig. 21, the initial waypoint commanded by the Team Planner for each rover lies outside of the rover FOV (i.e., ''90° clockwise'') and, upon start-up of this experiment, requires the rover to drive into unknown regions of the map. Prior to running the nonlinear optimization routine, the rover executes a turn-in-place to orient its camera FOV with the waypoint (Fig. 21) and then resumes running the trajectory optimization. Fig. 20 reflects this turn-in-place behavior through the twist command profiles, where the regions in red indicate when the rover issues a zero linear velocity command and a maximum angular velocity command. The spot turn continues until the waypoint falls within the rover FOV or the spot turn can terminate early if: 1) an obstacle appears within the rover radius footprint or 2) the rover has turned 90° (this prevents rover wheel dig-in into Lunar regolith).

This behavior was generally observable during the initial periods of an exploration or formation driving and also intermittently through the course of a drive. As such, we also employed a spot turn behavior to overcome the nonholonomic driving constraints for the rover. Although the inclusion of LiDAR for future missions might obviate the need for a spot turn behavior, we found in practice that this feature was crucial in improving system performance without the need for a more complex ''perception-aware'' cost to imbue into the nonlinear optimization routine.

FIGURE 22. **Testing was conducted to evaluate the efficacy of the planner in the presence of negative obstacles. A divert around a wide crater ($r > 2$ m) is shown at the top; a smaller crater ($r \approx 10$ cm) is shown at the bottom.**

4) CASE STUDY: NEGATIVE OBSTACLES

Testing was also conducted to assess the performance of the planner in the presence of negative obstacles (e.g., craters) and address specific concerns regarding the performance of the Local Planner when negative obstacles created large unknown regions. As shown in Fig. 22, we added ditches and craters of various sizes (approximately 10 cm to over 1 m in radius) into our outdoor testing environment. The largest of these craters induced large regions of unknown space in the traversability maps and is particularly notable, shown in the top half of Fig. 22. Here, a waypoint was commanded inside the crater; the rover traversed to the crater but as noted in Section V-A3 obtained an optimistic view of the map for the Global Planner. However, the unobserved crater region prevented the Local Planner from entering the unmapped steep terrain, shown in the path traversed around the edge of the crater. This serves as practical verification of the protective quality of the two-tiered planner utilizing different map treatments. Eventually, the rover timed out on its attempt to reach the waypoint, returning to an IDLE state. Smaller craters, as shown at the bottom of Fig. 22, are easily avoided over the Local Planner's planning horizon.

5) CASE STUDY: REPRESENTATIVE TERRAIN SPECIFICATION

Multiple 15-m drives were performed amid positive and negative obstacles similar to an aggressive terrain specification above that at the anticipated landing location in Reiner Gamma. These drives sought to challenge the rover with terrains with a large number of obstacles, long diverts, local minima, and occluded obstacles (e.g., obstacle hidden behind tall positive obstacles). Fig. 23 details one of these drives at left, including tall occluding rocks and a variety of rocks at radii approximately from the anticipated terrain specification. Global replans were successfully triggered as occluded regions came within view, including challenging ''dead end'' cases that require meter-long replanning. We note that these cases use the simple heuristic initialization strategy for long-distance driving, similar to the exploration driving test case. Shown at right is a dead-end scenario that

FIGURE 23. Long drives of up to 15 m in CFA-specification terrains to test long-horizon planning performance in aggressively cluttered terrain. The rover was able to successfully navigate many of these terrains. However, local minima such as box-canyon solutions shown to the right are possible without a sophisticated global initialization strategy.

was only observed once the rover was fully surrounded by collision geometry. Even in many of these cases, the rover was able to successfully inform new initialization to continue progress.

VIII. DISCUSSION

In this section, we review the limitations of CRESCENT as exposed during our hardware testing campaign, adjustments made to overcome some of these challenging cases, and future directions of investigation to improve the surface rover autonomy pipeline.

A. CHALLENGING CASES

1) LONG SHADOWS

As illustrated in Fig. 24, shadows consistently posed issues for the performance of the autonomy pipeline due to the use of stereo cameras as the primary perception system. On the Lunar surface, shadows can persist for multiple Earth days during the early morning and late afternoon periods of the approximately two-week Lunar day. Shadowed scenarios were replicated during nighttime testing under floodlights in the JPL Mars Yard. Similar to Lunar shadows, these nighttime shadows are very dark and featureless, with high contrast against surrounding lit terrain. In instances where shadows were cast in the path of the rover, the subsequent holes in the stereo pipeline cause mapping cells to be marked as unknown and thereby cause the Local Planner to treat shadows as obstacles. Additionally, long shadows during Lunar day and night can create large obstacles that extend beyond the Local Planner's designated horizon length. Although the use of LiDAR has been investigated to address this challenge for Lunar driving [64], a future area of research that investigates assigning semantic labels to parts of the map using a machine learning pipeline might address this issue of shadows while only using stereo cameras.

FIGURE 24. Nighttime testing conducted at the JPL Mars Yard exposed issues with shadows and the subsequent effect of creating large unknown regions for the Local Planner to treat as an obstacle.

2) OBSTACLES IN ROVER FOOTPRINT

Due to the placement of the rover stereo images, a commonly encountered issue was the appearance of unknown space within the rover footprint when the computed disparity image featured holes. Consequently, the behavior of the Local Planner to treat unknown cells as obstacles yields persistent planner failures. Although refinements to the Mapping component parameters addressed most instances of this phenomenon, future work might consider the use of learning-based or heuristic approaches to address holes in the disparity map.

3) LARGE CRATERS

Akin to the issue of long shadows, large craters posed a unique challenge as they appeared as unknown space in our mapping pipeline. Due to the placement of the stereo cameras on the rovers, the interior of large craters cannot be perceived by the cameras and caused issues when the rover was commanded to drive through a crater. As discussed in Section VII, we also sought to simulate the physical features of crater rims that are visible for many craters on the surface of the Moon. These crater rims were occasionally detected as obstacles by our mapping algorithm and this further exacerbated issues in planning around larger craters. For flight, we anticipate that crater rims will not pose a problem in practice by incorporating conservatism into the maps that are used to generate waypoints during the exploration and formation-driving phase and disallowing plan requests to individual agents that traverse through possible craters. However, addressing the issue of planning around large craters is more salient for future long-range Lunar driving campaigns, and avoiding craters altogether may lead to planning failures.

4) HIGHLY CLUTTERED ENVIRONMENTS

As discussed in Section VII, highly cluttered environments pose a challenge due to an excessive number of local minima and binary decision choices. While the real-time aspect for trajectory optimization is aided by online signed distance function computation, cluttered environments demand a

sophisticated global initialization strategy. Our work handles this aspect by relying on a sampling-based long-horizon planner (formation-driving Team Planner) and simple heuristics in between desired waypoints in other cases (exploration). CRESCENT's ability to maintain feasible plans until significant plan improvement is observed also helps to prevent switching homotopy classes between local minima.

B. AREAS OF IMPROVEMENT

Our proposed CRESCENT planner represents a significant advancement in the state-of-the-art for onboard planning for Lunar rover missions as compared to recent efforts such as the VIPER [8] and Pragyan Lunar rover [7]. As such, we developed a numerical optimization-based planner that fulfills the mission needs for CADRE, explicitly deals with mapping uncertainty, and introduces novelties in its formulation for real-time use, including a signed distance function factor graph constraint. Here, we identify lessons drawn from the verification and validation (V&V) testing for CADRE, to expand on areas for improvement which do not impede CADRE's mission but can enhance lunar planning for future missions. Furthermore, as highlighted through the substantial progress in autonomy demonstrated between successive Mars rover missions [14], we believe that numerical optimization-based approaches provide the flexibility to neatly encapsulate the following areas of improvement.

1) AGENT INTENT PREDICTION

In the CADRE concept of operations, the Team Planner operates in a centralized fashion to generate planning requests for each individual agent planner, but the motion planner used by individual rovers is not cognizant of the remaining agents. Consequently, we observed cases during which a rover would enter the FOV of another rover and be detected as a hazard. Although the motion planners behaved as expected to avoid collision with another rover, there were instances wherein a detected rover would exit the FOV and change the homotopy class of the determined plan, thereafter possibly timing out the spatiotemporal constraint during formation driving due to homotopy class jumps. Instead, a future direction of work might encapsulate information about other agents within the Global Planner by using a mapping pipeline that predicted neighboring agent behaviors and incorporated these forecasts as collision avoidance constraints for dynamic obstacles.

2) HANDLING UNCERTAINTY

Our motion planning pipeline disregards uncertainty and plans in a wholly deterministic fashion. Although replanning in a receding horizon fashion allows the system to correct for disturbances, a promising area of future research is to imbue CRESCENT with an explicit notion of uncertainty stemming from the state estimation and dynamics pipeline. Namely, a nascent area of research has been to develop uncertainty-aware trajectory optimization formulations [65], [66] and these approaches propagate uncertainty within the optimal control problem to generate robust plans and control actions. Although these formulations typically introduce new decision variables that increase the size of the nonlinear optimization problem, recent approaches proposing tailored stochastic optimal control solvers show promise for being able to run in real time on embedded platforms [67] and stand to improve the safety of the rover decision-making capabilities.

3) ONLINE ADAPTATION

The kinematics model used in our trajectory optimization framework uses a simple unicycle model [68] to enforce the vehicle driving constraints. Although such an analytical model provides a parsimonious and interpretable representation of the rover driving behavior, these analytical models may markedly differ from the driving conditions encountered during the mission due to, e.g., hardware degradation experienced through the concept of operations, changing terramechanics conditions across environments, among others. As such, future planning and control paradigms should be equipped with the capability to adapt to unforeseen and changing driving conditions and a promising direction of work is to infuse physics-inspired models with data-driven approaches that meld information gleaned from online sensor data. Recent works have demonstrated the promise of interleaving such data-driven approaches with trajectory optimization tasks [69], [70] and such approaches could allow for expanding the operating envelope for autonomous motion planners.

IX. CONCLUSION

In this work, we presented the development of an optimization-based motion planning algorithm for use in the upcoming CADRE Lunar rover mission and highlights of its verification and validation testing. Our proposed CRESCENT approach represents a major advancement in terms of online trajectory generation and motion planning capabilities compared to the state-of-the-art for surface rover autonomy capabilities. By formulating the motion planning problem as an optimal control problem, we leveraged off-the-shelf nonlinear optimization solvers to efficiently and robustly satisfy the challenging spatiotemporal and safety constraints necessary for accomplishing the science objectives of the CADRE mission. We also constructed novel constraint formulations using signed distance function factors to ensure real-time computation on under a strict embedded compute budget, and developed strategies for roving with incomplete map information on the Lunar surface. We developed and tuned our proposed CRESCENT planner in a simple kinematic simulation environment and through representative environment distributions as a part of a Monte Carlo testing pipeline. We then rigorously validated the performance of the planner through hardware testing conducted at the JPL Mars and Mini-Mars Yard test facilities.

ACKNOWLEDGMENT

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004). Government sponsorship acknowledged. This work was completed while Abhishek Cauligi and Keenan Albee were with the CADRE GNC Team at the Jet Propulsion Laboratory, California Institute of Technology. The authors would like to thank Dustin Aguilar, Libby Boroson, Nihal Dhamani, Grace Lim, Pedro Proença, Adit Sahasrabudhe, Patrick Spieler, and the entirety of the CADRE V&V and GNC Team for their feedback and suggestions during the development of this work.

REFERENCES

[1] Y. Gao and S. Chien, "Review on space robotics: Toward top-level science through space exploration," *Sci. Robot.*, vol. 2, no. 7, pp. 1–11, Jun. 2017.

[2] A. Rankin, M. Maimone, J. Biesiadecki, N. Patel, D. Levine, and O. Toupet, "Driving curiosity: Mars rover mobility trends during the first seven years," in *Proc. IEEE Aerosp. Conf.*, Mar. 2020, pp. 1–19.

[3] J. A. Starek, B. Açıkmeşe, I. A. Nesnas, and M. Pavone, "Spacecraft autonomy challenges for next-generation space missions," in *Advances in Control System Technology for Aerospace Applications*. Cham, Switzerland: Springer, 2016, ch. 1.

[4] L. Werner, P. Proença, A. Nüchter, and R. Brockers, "Covariance based terrain mapping for autonomous mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2024, pp. 11768–11773.

[5] J.-P. de la Croix et al., "Multi-agent autonomy for space exploration on the CADRE lunar technology demonstration," in *Proc. IEEE Aerosp. Conf.*, Mar. 2024, pp. 1–14.

[6] *2024 NASA Civil Space Shortfall Rankings*, NASA, Washington, DC, USA, 2024.

[7] R. Ghosh, S. Tomar, C. S. Mhatre, K. Sumithra, B. G. V. P. Kumar, and M. S. Siva, "Path planning for the pragyan rover: Experiences and challenges," in *Proc. Int. Conf. Space Robot. (iSpaRo)*, Jun. 2024, pp. 70–75.

[8] S. Banerjee, E. Balaban, M. Shirley, K. Bradner, and M. Pavone, "Contingency planning using bi-level Markov decision processes for space missions," in *Proc. IEEE Aerosp. Conf.*, Mar. 2024, pp. 1–9.

[9] J. D. Baker et al., "The endurance lunar rover sample return mission," in *Proc. IEEE Aerosp. Conf.*, Mar. 2024, pp. 1–13.

[10] J. Schonfeld, "Summary of the contracted deliveries of NASA payloads to the moon via commercial lunar payload services (CLPS)," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2023, pp. 863–866.

[11] S. Nayak, M. Paton, and M. W. Otte, "A heuristic-guided dynamical multi-rover motion planning framework for planetary surface missions," *IEEE Robot. Autom. Lett.*, vol. 8, no. 5, pp. 2542–2549, May 2023.

[12] M. Saboia, F. Rossi, V. Nguyen, G. Lim, D. Aguilar, and J.-P. de la Croix, "CADRE MoonDB: Distributed database for multi-robot information-sharing and map-merging for Lunar exploration," in *Proc. Int. Workshop Auto. Agents Multi-Agent Syst. Space Appl.*, 2024, pp. 1–9.

[13] O. Toupet, T. Del Sesto, M. Ono, S. Myint, J. vander Hook, and M. McHenry, "A ROS-based simulator for testing the enhanced autonomous navigation of the Mars 2020 rover," in *Proc. IEEE Aerosp. Conf.*, Mar. 2020, pp. 1–11.

[14] V. Verma et al., "Autonomous robotics is driving perseverance rover's progress on Mars," *Sci. Robot.*, vol. 8, no. 80, pp. 1–12, Jul. 2023.

[15] O. Toupet, M. Ono, T. D. Sesto, M. Maimone, and M. McHenry, "Enhanced autonomous navigation on the perseverance Mars rover," *IEEE Trans. Field Robot.*, early access, Nov. 24, 2025, doi: 10.1109/TFR.2025.3636366.

[16] *Origins, Worlds, and Life: A Decadal Strategy for Planetary Science and Astrobiology 2023–2032*, National Academies of Sciences, Engineering, and Medicine, Washington, DC, USA, 2022.

[17] J. T. Betts, "Survey of numerical methods for trajectory optimization," *J. Guid., Control, Dyn.*, vol. 21, no. 2, pp. 193–207, Mar. 1998.

[18] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Rev.*, vol. 59, no. 4, pp. 849–904, Jan. 2017.

[19] L. Blackmore, "Autonomous precision landing of space rockets," *Bridge*, vol. 46, no. 4, pp. 15–20, 2016.

[20] D. Malyuta, Y. Yu, P. Elango, and B. Açıkmeşe, "Advances in trajectory optimization for space vehicle control," *Annu. Rev. Control*, vol. 52, pp. 282–315, Jan. 2021.

[21] K. Albee, M. Ekal, B. Coltin, R. Ventura, R. Linares, and D. W. Miller, "The RATTLE motion planning algorithm for robust online parametric model improvement with on-orbit validation," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 10946–10953, Oct. 2022.

[22] A. V. Rao, "A survey of numerical methods for optimal control," *Adv. Astron. Sci.*, vol. 135, no. 1, pp. 497–528, 2010.

[23] C. Mastalli et al., "Crocoddyl: An efficient and versatile framework for multi-contact optimal control," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 2536–2542.

[24] A. Domahidi, E. Chu, and S. Boyd, "ECOS: An SOCP solver for embedded systems," in *Proc. Eur. Control Conf. (ECC)*, Jul. 2013, pp. 3071–3076.

[25] G. Frison and M. Diehl, "HPIPM: A high-performance quadratic programming framework for model predictive control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, 2020.

[26] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs," *Math. Program. Comput.*, vol. 12, no. 4, pp. 637–672, Dec. 2020.

[27] A. Bambade, S. El-Kazdadi, A. Taylor, and J. Carpentier, "PROX-QP: Yet another quadratic programming solver for robotics and beyond," in *Proc. Robot., Sci. Syst. XVIII*, Jun. 2022, pp. 1–12.

[28] P. Elango, A. G. Kamath, Y. Yu, B. Acikmese, M. Mesbahi, and J. M. Carson, "Correction: A customized first-order solver for real-time powered-descent guidance," in *Proc. AIAA SCITECH Forum*, Jan. 2022, pp. 1–14.

[29] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 1917–1922.

[30] J. Schulman et al., "Motion planning with sequential convex optimization and convex collision checking," *Int. J. Robot. Res.*, vol. 33, no. 9, pp. 1251–1270, Aug. 2014.

[31] D. Morgan, S.-J. Chung, and F. Y. Hadaegh, "Model predictive control of swarms of spacecraft using sequential convex programming," *J. Guid., Control, Dyn.*, vol. 37, no. 6, pp. 1725–1740, Nov. 2014.

[32] R. Bonalli, A. Cauligi, A. Bylard, and M. Pavone, "GuSTO: Guaranteed sequential trajectory optimization via sequential convex programming," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 6741–6747.

[33] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, Mar. 2019.

[34] R. Verschueren et al., "Acados—A modular open-source framework for fast embedded optimal control," *Math. Program. Comput.*, vol. 14, no. 1, pp. 147–183, Mar. 2022.

[35] D. Malyuta et al., "Convex optimization for trajectory generation: A tutorial on generating dynamically feasible trajectories reliably and efficiently," *IEEE Control Syst. Mag.*, vol. 42, no. 5, pp. 40–113, Oct. 2022.

[36] T. A. Howell, B. E. Jackson, and Z. Manchester, "ALTRO: A fast solver for constrained trajectory optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 7674–7679.

[37] K. Nguyen, S. Schoedel, A. Alavilli, B. Plancher, and Z. Manchester, "TinyMPC: Model-predictive control on resource-constrained microcontrollers," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2024, pp. 1–7.

[38] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Trans. Control Syst. Technol.*, vol. 18, no. 2, pp. 267–278, Mar. 2010.

[39] C. Rösmann, F. Hoffmann, and T. Bertram, "Timed-elastic-bands for time-optimal point-to-point nonlinear model predictive control," in *Proc. Eur. Control Conf. (ECC)*, Jul. 2015, pp. 3352–3357.

[40] L. Tiziani, Y. Zhang, F. Dellaert, and F. L. Hammond, "Factor graph-based trajectory optimization for a pneumatically-actuated jumping robot," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 9870–9876.

[41] H. Martiros et al., "SymForce: Symbolic computation and code generation for robotics," in *Proc. Robotics: Sci. Syst.*, Jun. 2022, pp. 1–12.

[42] L. Villaseñor-Pineda et al., "Theseus: A library for differentiable nonlinear optimization," in *Proc. Conf. Neural Inf. Process. Syst.*, 2022, pp. 1–18.

[43] R. Soussan, V. Kumar, B. Coltin, and T. Smith, "AstroLoc: An efficient and robust localizer for a free-flying robot," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 4106–4112.

[44] T. P. Setterfield, R. A. Hewitt, A. T. Espinoza, and P.-T. Chen, "Feature-based scanning LiDAR-inertial odometry using factor graph optimization," *IEEE Robot. Autom. Lett.*, vol. 8, no. 6, pp. 3374–3381, Jun. 2023.
[45] M. Kobilarov, "Cross-entropy motion planning," *Int. J. Robot. Res.*, vol. 31, no. 7, pp. 855–871, Jun. 2012.
[46] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 1433–1440.
[47] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.
[48] M. Pivtoraiko, D. Mellinger, and V. Kumar, "Incremental micro-UAV motion replanning for exploring unknown environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 2452–2458.
[49] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "PAMPC: Perception-aware model predictive control for quadrotors," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 1–8.
[50] C. Rösmann, F. Hoffmann, and T. Bertram, "Kinodynamic trajectory optimization and control for car-like robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 5681–5686.
[51] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robot. Auto. Syst.*, vol. 88, pp. 142–153, Feb. 2017.
[52] G. Bouza and G. Still, "Mathematical programs with complementarity constraints: Convergence properties of a smoothing method," *Math. Operations Res.*, vol. 32, no. 2, pp. 467–483, May 2007.
[53] J. Carius, R. Ranftl, V. Koltun, and M. Hutter, "Trajectory optimization with implicit hard contacts," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3316–3323, Oct. 2018.
[54] B. Aceituno-Cabezas et al., "Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2531–2538, Jul. 2018.
[55] A. Cauligi, A. Chakrabarty, S. Di Cairano, and R. Quirynen, "PRISM: Recurrent neural networks and presolve methods for fast mixed-integer optimal control," *Learn. Dyn. Control*, vol. 168, pp. 34–46, Jun. 2022.
[56] T. Marcucci, P. Nobel, R. Tedrake, and S. Boyd, "Fast path planning through large collections of safe boxes," *IEEE Trans. Robot.*, vol. 40, pp. 3795–3811, 2024.
[57] M. King–Smith, P. Tsiotras, and F. Dellaert, "Simultaneous control and trajectory estimation for collision avoidance of autonomous robotic spacecraft systems," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 257–264.
[58] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed., Cham, Switzerland: Springer, 2006.
[59] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G^2o: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 3607–3613.
[60] M. W. Smith et al., "On-orbit results and lessons learned from the ASTERIA space telescope mission," in *Proc. AIAA/UTU Small Satell. Conf.*, 2018, pp. 1–20.
[61] A. Rizvi, K. Ortega, and Y. T. He, "Developing lunar flashlight and near-Earth asteroid scout flight software concurrently using open-source f prime flight software framework," in *Proc. AIAA/UTU Small Satell. Conf.*, 2025, pp. 1–10.
[62] M. Golombek and D. Rapp, "Size-frequency distributions of rocks on Mars and Earth analog sites: Implications for future landed missions," *J. Geophys. Res., Planets*, vol. 102, no. E2, pp. 4117–4129, Feb. 1997.
[63] R. Bonalli, A. Cauligi, A. Bylard, T. Lew, and M. Pavone, "Trajectory optimization on manifolds: A theoretically-guaranteed embedded sequential convex programming approach," in *Proc. Robot., Sci. Syst. XV*, Jun. 2019, pp. 1–10.
[64] S. Daftry et al., "LunarNav: Crater-based localization for long-range autonomous lunar rover navigation," in *Proc. IEEE Aerosp. Conf.*, Mar. 2023, pp. 1–15.
[65] K. Albee et al., "A robust observation, planning, and control pipeline for autonomous rendezvous with tumbling targets," *Frontiers Robot. AI*, vol. 8, Sep. 2021, pp. 1–21.
[66] K. Echigo, A. Cauligi, and B. Açıkmeşe, "Expected time-optimal control: A particle model predictive control-based approach via sequential convex programming," in *Proc. IEEE 63rd Conf. Decis. Control (CDC)*, Dec. 2024, pp. 1430–1436.
[67] A. Lahr, A. Zanelli, A. Carron, and M. N. Zeilinger, "Zero-order optimization for Gaussian process-based model predictive control," *Eur. J. Control*, vol. 74, pp. 1–9, Nov. 2023.
[68] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.
[69] S. Banerjee, J. M. Harrison, P. M. Furlong, and M. Pavone, "Adaptive meta-learning for identification of rover-terrain dynamics," in *Proc. Int. Symp. Artif. Intell.*, 2020, pp. 1–8.
[70] R. Majumdar, D. C. Sternberg, K. Albee, and O. Jia-Richards, "Demonstration of the dyna reinforcement learning framework for reactive close proximity operations," in *Proc. AIAA SCITECH Forum*, Jan. 2025, pp. 1–16.