A Heuristic-Guided Dynamical Multi-Rover Motion Planning Framework for Planetary Surface Missions

Sharan Nayak, *Graduate Student Member, IEEE*, Michael Paton, and Michael W. Otte, *Member, IEEE*

***Abstract*—We present a heuristic-guided multi-robot motion planning framework that solves the problem of *n* dynamical agents visiting *m* unlabeled targets in a partially known environment for planetary surface missions without solving the two-point boundary value problem (BVP). The framework design is motivated by typical planetary surface mission constraints of limited power, limited computation, and limited communication. The framework maintains a centralized, dynamically updated probabilistic roadmap (PRM) that incorporates new obstacle updates as the agents move in the environment. The dynamic roadmap captures the changing obstacle topology and provides updated cost-to-go heuristics to accelerate each agent's independent single-query motion-planning process. The agents use a feasible sampling-based motion planner without computing the BVP while leveraging the roadmap heuristics to quickly plan and visit their assigned target. The agents handle robot-robot and robot-obstacle collision avoidance in a decentralized fashion. We conduct multiple simulation experiments using robots with non-linear dynamics to show our planner performs better in overall planning time and mission time than approaches not using the roadmap heuristic. We also field our algorithm on prototype rovers and demonstrate the viability of implementing our algorithm on real-world hardware platforms.**

***Index Terms*—Path planning for multiple mobile robots, motion planning, and dynamics.**

I. Introduction

MULTI-ROBOT teams have been proposed for planetary surface exploration missions. These missions typically require team members to coordinate, plan motions, and move to specific locations of interest (targets) in communication-restricted environments. Often, these robots have partial or no information about the environment at the mission onset. Moreover, the robots may have constrained dynamics making it hard or impossible to compute the two-point boundary value problem (BVP) solution [1] for their motion planning. In this work, we propose a multi-robot feasible motion planning framework that considers partially known planetary surface environments and complex robot dynamics to coordinate robots to visit multiple unlabeled targets without solving the two-point BVP. The framework choice is motivated by limited power, computation and communication – constraints that are typical for a multi-robot planetary surface mission.

Using robot teams for planetary surface exploration is gaining popularity. Recently, NASA announced the Cooperative Autonomous Distributed Robotic Explorers (CADRE) technology demonstration [2] that proposes using robot teams to explore the lunar surface. Prior terrestrial information (e.g., low-resolution images) obtained from the orbiting Lunar Reconnaissance Orbiter (LRO) can be used to accelerate robot motion planning by providing information about obstacles.

Probabilistic Roadmaps (PRM) [3] sample valid configurations in the robot's free space and use trajectories that respect the robot's dynamics to connect them to form motion graphs, which are useful for path planning. Computing a dynamical trajectory that exactly connects two sampled nodes requires a solution to the two-point BVP. However, computing the two-point BVP for robots with complex dynamics is challenging, and closed-form solutions are often hard to come by. Nevertheless, previous work [4], [5] has shown that it is possible to use roadmaps computed offline and built in a lower-dimensional subspace of the state space (where the two-point BVP can be solved) to speed up online motion planning.

In our work, we have a central entity called the *base station* that maintains a dynamic roadmap that is updated as new obstacles information is received from the robots. The base station model makes sense in lander missions where robots remain in the lander vicinity; in other cases, any robot can take the role of the central entity. Motion planning on each individual robot is accelerated by using focusing heuristics derived from the base station's roadmap (Fig. 1).

With multi-robot motion planning, there is also the question of which entity computes each robot's plans. Many previous works have focused on a centralized approach [6], [7] where a central entity calculates each agent's motion plan. The problem with this approach is that the central entity has to transmit the whole path to each agent at the cost of additional communication. Moreover, all robots must wait for the central entity to send them a path before they are able to move. In our work, we design a *hybrid centralized-decentralized* approach where the target assignment is done on the base station, yet each robot calculates its own path to the target (using the roadmap heuristic), removing the need to send 'path' messages and leading to a better distribution of computational load among the individual robots. It is possible to make our approach fully decentralized by having

Manuscript received 10 October 2022; accepted 18 February 2023. Date of publication 8 March 2023; date of current version 20 March 2023. This letter was recommended for publication by Associate Editor L. Pimenta and Editor M. A. Hsieh upon evaluation of the reviewers' comments. The work of Sharan Nayak was supported by Clark Doctoral Fellowship fund. The work of Michael Otte and Sharan Nayak was supported by Minta Martin Research Fund (UMD). This work was supported by Jet Propulsion Laboratory (JPL), California Institute of Technology, through contract with the National Aeronautics and Space Administration under Grant 80NM0018D0004. *(Corresponding author: Michael Paton.)*

Sharan Nayak and Michael W. Otte are with the A. James Clark School of Engineering, Aerospace Engineering Department, University of Maryland, College Park, MD 20742 USA (e-mail: snayak18@umd.edu; otte@umd.edu).

Michael Paton is with the JPL, California Institute of Technology, Pasadena, CA 91109 USA (e-mail: michael.paton@jpl.nasa.gov).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2023.3254444, provided by the authors.

Digital Object Identifier 10.1109/LRA.2023.3254444

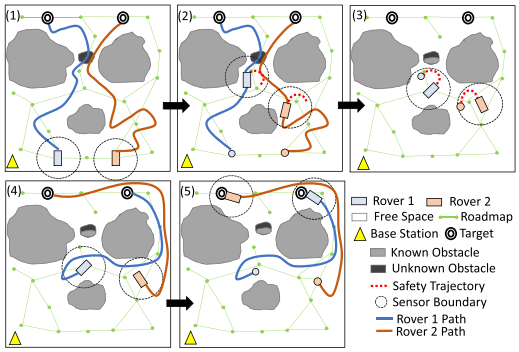

Fig. 1. (1) A base station creates a centralized roadmap considering known obstacles and then assigns targets to rovers. Rovers compute feasible paths to their assigned targets using the roadmap focusing heuristic as a guide without computing the two-point BVP. (2) Rover 1 detects an unknown obstacle and informs Rover 2 (through the base station). Both compute safety trajectories to come to rest. (3) Base station updates the roadmap and its heuristics to account for the new obstacle. (4) Rovers find paths to their newly assigned targets using the updated heuristic. (5) Rovers reach their assigned targets.

each agent maintain its own copy of the roadmap. However, the decentralized approach usually involves more communication links (fully connected (n^2)) than the centralized approach (star (n)).

The letter's main contribution is a heuristic-guided multi-robot feasible motion planning framework that solves the problem of n robots (with dynamics) visiting m unlabeled target locations in a partially known environment for planetary-surface missions. The word *unlabeled* implies that it does not matter which robot visits what target as long as all targets are visited. Additional contributions include:

1) Simulation experiments using three different scenarios and two dynamical systems to show that our planner improves overall planning time and mission time than approaches not using a roadmap heuristic.
2) Implementation on prototype rovers at NASA JPL Lunar Mini-yard to show the viability of using our proposed approach on real-world hardware.
3) An analysis of the overall planning time variation vs. number of initial PRM nodes for two dynamical systems.
4) A scalability experiment analysis showing the variation of total planning time and mission time with increase in the number of robots and targets.

The rest of the article is organized as follows. Section II discusses related work. Section III contains the preliminaries. Section IV provides a formal definition of our work. Section V discusses our proposed motion planning framework. Section VI describes the setup used for running experiments. Section VII presents the experiment results. Section VIII concludes by summarizing the contributions and main results.

II. Related Work

A. Roadmap-Based Methods for Multi-Robot Motion Planning

Švestka et al. [8] use a centralized approach for roadmap-based multi-robot path planning. They plan on an explicit higher-dimensional composite roadmap created by taking the cartesian product of individual robot roadmaps. Wagner et al. [9] improve this work by showing it is sufficient to use an efficient implicit composite roadmap representation and apply their M* algorithm [10] to calculate paths. dRRT [11] and dRRT* [12] further use this implicit representation to perform an RRT-like search in discrete space to find robot paths efficiently. But, these approaches do not work for complex robot dynamics when the two-point BVP cannot be solved.

Van der Berg and Overmars [13] use a local planner that encodes dynamic constraints while employing A* to calculate global paths on a pre-computed roadmap. They use a prioritized planning scheme where each robot is assigned a priority, with motion planning prioritized for robots with decreasing priority. Le and Plaku introduced various centralized approaches for multi-robot motion planning for labeled [6], [7], [14], and unlabeled goals [15] while considering robot dynamics. They build a motion tree over the composite state space of all robots while using a lower-dimensional roadmap for each robot to guide the search. The completed motion tree's trajectories correspond to each robot's feasible paths. However, these approaches assume a fully known environment at the start of the mission. In contrast, our work considers partially known settings where new obstacles are detected as robots move in the environment.

B. Multi-Robot Path Planning in Dynamic Environments

Prior work has employed disparate approaches to handle dynamic environments. The definition of a "dynamic" environment varies in the literature but is usually characterized by the encounter of another robot or an unexpected obstacle, unknown during initial planning. Shwail et al. [16] construct a roadmap in the preprocessing step and use A* to find paths for robots in an offline phase. In the online phase, after each delta time step, a collision check with other robots is performed, and the robot executes a bang-bang control along with a wait option to avoid other robots and reach the goal position. Tordesillas and How [17] present MADER, where the trajectories of robots are characterized by an outer polyhedral surrounding them. MADER introduces linear separability between polyhedra using planes in an optimization problem for collision avoidance with other robots and obstacles.

A whole body of work has focused on using machine learning strategies. Zhu et al. [18] use a recurrent neural network to learn multi-robot motion behaviors from demonstrated trajectories. The learned trajectory model is used in conjunction with a model predictive controller (MPC) to perform robot-robot and robot-obstacle collision avoidance. PRIMAL [19] uses reinforcement and imitation learning to learn decentralized robot policies for robot and obstacle collision avoidance that can extend to any number of robots. PRIMAL 2 [20] extends this work to lifelong planning, where robots visit other goals upon reaching their assigned goal. However, these machine learning strategies do not encode robot dynamics while planning.

Our multi-robot motion planning framework differs from previous work in that we consider both robot dynamics and dynamic environments (unknown static obstacles and moving robots) limited by planetary exploration mission constraints.

III. Preliminaries

Let the workspace $\mathcal{W}$ contain n robots and m unlabeled target positions with no relative constraints between n and m. Let the state-space of robot i be denoted by $\mathcal{X}_i$. Let $\mathcal{U}_i$ be its input control space. Let its obstacle space, $\mathcal{X}_{obs_i}$, be an open subset of $\mathcal{X}_i$. $\mathcal{X}_{obs_i}$ is the set of all states where the system is in

collision with obstacles. The free space $\mathcal{X}_{free_i}$ is defined as $\mathcal{X}_{free_i} = \mathcal{X}_i \backslash \mathcal{X}_{obs_i}$. $\mathcal{X}_{free_i}$ is the closed subset of $\mathcal{X}_i$ and the set of all states that are not defined by obstacle collisions. The free space changes as agents move in the environment and detect new obstacles and receive new obstacle updates from each other via the base station. Because of this, we make a simplifying assumption that $\mathcal{X}_{free_i}$ is a function of time, i.e., $\mathcal{X}_{free_i}(t)$. We assume that $\mathcal{X}_{free_i}(t)$ changes unpredictably and cannot be estimated from its earlier states. $\mathcal{X}_{free_i}(t)$ does not have smooth properties in general, as the obstacle changes may be instantaneous. We assume the robot's sensor radius to be at least $2 * d_{stop}$ where d_{stop} is the robot's maximum stopping distance. d_{stop} is multiplied by two to account for two robots heading toward each other with maximum velocity.

The robot system dynamics f_i satisfies the differential equation of the form:

$$\dot{x_i}(t) = f_i\left(x_i(t), u_i(t)\right), \quad x_i(t) \in \mathcal{X}_i, \quad u_i(t) \in \mathcal{U}_i. \tag{1}$$

Collision-free planning involves

$$x_i(t) \in \mathcal{X}_{free_i}(t) \;\; \wedge \;\; \texttt{OC}(\texttt{x}_\texttt{i}(\texttt{t})) \neq \texttt{OC}(\texttt{x}_\texttt{j}(\texttt{t})), \; \forall j, \forall t \tag{2}$$

where $\texttt{OC}$ is the robot's occupied volume. Let x_1 and x_2 in $\mathcal{X}_i$. Then a trajectory $\mathcal{E}_i(x_1, x_2)$ connecting x_1 and x_2 is defined as the continuous mapping $\mathcal{E}_i : [0, 1] \rightarrow \mathcal{X}_{free_i}(t)$ such that $\mathcal{E}_i(0) = x_1$ and $\mathcal{E}_i(1) = x_2$. Although the terms path and trajectory have been used interchangeably in the literature, we define a path π_i as the concatenation of a sequence of trajectories, i.e. $\pi_i \triangleq \mathcal{E}_{i_1} \oplus \mathcal{E}_{i_2} \ldots \mathcal{E}_{i_l} \oplus \mathcal{E}_{i_{l+1}} \ldots$ such that $\mathcal{E}_{i_l}(1) = \mathcal{E}_{i_{l+1}}(0)$, where $\oplus$ is the concatenation operator.

In the rest of the document, we assume homogeneous robots and all robots know the known obstacles i.e. $f_i = f_j, \mathcal{U}_i = \mathcal{U}_j = \mathcal{U}, \mathcal{X}_{free_i}(t) = \mathcal{X}_{free_j}(t) = \mathcal{X}_{free}(t) \, \forall i, j \leq n$.

IV. Problem Definition

Let $x_{s_i} \in \mathcal{X}_{free}(t)$ represent the ith robot's initial position at the start of the mission. Let x_{g_j} be the jth target (goal) and $\mathcal{X}_{g_j}$ its corresponding target region such that $x_{g_j} \in \mathcal{X}_{g_j}$, and $\mathcal{X}_{g_j} \subset \mathcal{X}_{free}(t)$. Let $\mathbf{S} = \{x_{s_1}, x_{s_2}, x_{s_3} \ldots x_{s_n}\}$ be the set of initial positions of the robots and $\mathbf{G} = \{\mathcal{X}_{g_1}, \mathcal{X}_{g_2}, \mathcal{X}_{g_3} \ldots \mathcal{X}_{g_m}\}$ be the set of target (goal) regions. Let T_{comp} represent the time taken to complete the mission, i.e., visit all targets.

Let robot i be initially assigned a target and let π_{i_1} be the corresponding path planned. As robot i executes π_{i_1}, it may require a replan due to changes in $\mathcal{X}_{free}(t)$ or when it reaches the assigned target. During replan, it may be assigned the same target, a different target, or no target. Let π_{i_2} be the new path planned to reach its next assigned target. Furthermore, let $\pi_{c_i} \triangleq \pi_{i_1} \oplus \pi_{i_2} \ldots \pi_{i_l} \oplus \pi_{i_{l+1}} \ldots$ represent the combined feasible path executed by robot i from mission start to end where $\pi_{i_l}(1) = \pi_{i_{l+1}}(0)$. Correspondingly, let $\pi_{c_1}, \pi_{c_2} \ldots \pi_{c_n}$ represent the complete paths executed by the robots $1, 2 \ldots n$ from the mission start to end, which includes visiting multiple targets. Let $\mathbf{G}_{\pi_c}$ denote the set of targets along π_c, i.e. $\mathbf{G}_{\pi_c} = \{\, \mathbf{G}_s \mid (\mathbf{G}_s \subseteq \mathbf{G} \;\wedge\; (\exists t \in [0, T_{comp}] \;\mid \pi_c(t) \in \mathbf{G}_s))\}$.

Problem: Multi-robot motion planning with unlabeled targets in a partially known environment.

Given ($\mathbf{S}$, $\mathcal{X}_{free}(t)$, $\mathbf{G}$)*, let robots plan and replan to accommodate changes in* $\mathcal{X}_{free}$ *and find feasible paths,* $\pi_{c_1}, \pi_{c_2} \ldots \pi_{c_n}$*, such that*

$$\mathbf{G}_{\pi_{c_1}} \cup \mathbf{G}_{\pi_{c_2}} \ldots \cup \mathbf{G}_{\pi_{c_n}} = \mathbf{G}$$

Multi-robot motion planning architecture

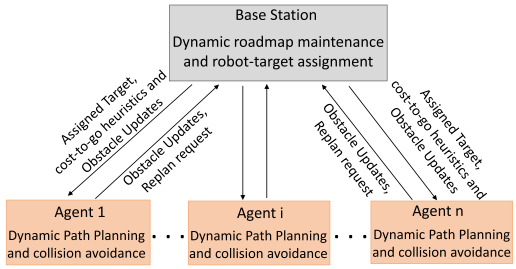

Fig. 2. The base station creates and maintains a dynamic roadmap and generates the robot-target assignment. The agent produces feasible paths and handles robot-obstacle and robot-robot collision avoidance. Communication only happens between the base station and agents and not among agents themselves.

Algorithm 1: `BaseStation` $(\mathbf{O}_{kwn}, \mathbf{S}, \mathbf{G}, N)$.

1 $\mathcal{G}_{rm} \leftarrow$ `ConstructRoadmap`$(\mathbf{O}_{kwn}, \mathbf{S}, \mathbf{G}, N)$
2 **while not** `AllTargetsReached` () **do**
3 **if** *Replan request by an agent* **then**
4 $\mathbf{S}_{cur} \leftarrow$ `CurrentRobotLocations`()
5 $\mathbf{O}_{new} \leftarrow$ `NewObstacles`()
6 $\mathcal{G}_{rm} \leftarrow$ `UpdateRoadMap`$(\mathcal{G}_{\texttt{rm}}, \mathbf{S}_{\texttt{cur}}, \mathbf{O}_{\texttt{new}})$
7 $\mathbf{H} \leftarrow$ `getHeuristics`$(\mathcal{G}_{rm})$
8 $\mathbf{A} \leftarrow$ `RobotTargetAssignment`$(\mathbf{S}_{\texttt{cur}}, \mathbf{G}, \mathbf{H})$
9 `SendTargetAndHeuristicsToRobots`$(\mathbf{A}, \mathbf{H})$
10 **if** `size`$(\mathbf{O}_{new}) > 0$ **then**
11 `InformOtherAgentsOfNewObstacles`$(\mathbf{O}_{new})$

V. Framework Description

The main idea behind our multi-robot sampling-based motion planning framework is to create and maintain a centralized dynamic roadmap to produce focusing heuristics that each agent can utilize to accelerate the feasible path creation to its assigned target without computing a two-point BVP solution. We update the roadmap and its cost-to-go heuristics when new obstacle updates are received from individual agents. We additionally handle the robot-target assignment and collision avoidance (robot-robot and robot-obstacle) to facilitate robots to visit all targets promptly without collisions.

In the following subsections, we present our framework components: i) Base-station computed dynamic roadmap maintenance and robot-target assignment (Algorithm 1), ii) Agent-computed feasible path planning and collision avoidance (Algorithms 2–6). A base station-agent communication layout can be seen in Fig. 2.

A. Framework Component That Runs on the Base Station

The base station (Algorithm 1) initially creates a probabilistic roadmap $\mathcal{G}_{rm} = (\mathbf{V}_{rm}, \mathbf{E}_{rm})$ in a lower-dimensional subspace of the robot's state space using a list of known obstacles $\mathbf{O}_{kwn}$ (known obstacles can be large obstacles detected through a prior map) (line 1). $\mathbf{V}_{rm}$ node list initially contains the robot's start and target configurations along with uniform randomly sampled configurations with $|\mathbf{V}_{rm}| = N$, where N is a user-input parameter that defines the initial number of roadmap nodes. $\mathbf{E}_{rm}$ includes the edges obtained by attempting to connect a roadmap node to its k nearest neighbors without collision. $\mathcal{G}_{rm}$ is dynamic [21] and gets updated (line 6) to account for new obstacles (e.g. obstacles detected by robot's sensors) as the mission progresses.

Algorithm 2: $\texttt{Agent}_\texttt{i}$ $(\mathcal{U}, \delta_{hr}, q)$.

```
1  while not AllTargetsReached() do
2      WaitUntilTargetAndHeuristicsReceived()
3      X_{g_i} ← getTargetAssignment()
4      H_j ← getHeuristics()
5      x_{start_i} ← getCurrentLocation()
6      π_i ← MotionPlan(x_{start_i}, X_{g_i}, H_j, U, δ_hr, q)
7      moveRobot ← True
8      while moveRobot do
9          MoveRobotAlongPath(π_i)
10         if Target reached then
11             BringRobotToRest()
12             moveRobot ← False
13             RequireReplanByBaseStation()
14         if Detect unknown obstacles within sensor radius then
15             InformNewObstaclesToBaseStation()
16             if ObstacleCollidesPath(π) then
                   // Execute safety trajectory
17                 BringRobotToRest()
18                 moveRobot ← False
19                 RequireReplanByBaseStation()
20         if Detect Robots within sensor radius then
               // Execute safety trajectory
21             BringRobotToRest()
22             π_i ← HandleCollisionWithOtherRobots()
```

Algorithm 3: $\texttt{HIRRT}(x_{start}, \mathcal{X}_{goal}, \mathbf{H}_{rm}, \mathcal{U}, \delta_{hr}, q)$.

```
1  G_tree ← {V_tree ← {x_start}, E_tree ← {}}
2  Q ← {}
3  for k ← 1 to M_iter do
       // c_rand - uniform random value
4      c_rand ~ U([0, 1])
5      E_tree = NULL
6      if c_rand < q then
7          x_pop ← Pop(Q)
8          if x_pop ≠ NULL then
               // Perform Exploitation
9              E_tree ← Exploit(H_rm, U, x_pop, δ_hr)
10     else
           // Perform Exploration
11         E_tree ← Explore(G_tree, U)
12     if E_tree ≠ NULL and NoCollision(E_tree) then
13         x_tree ← E_tree.finalNode()
14         E_tree ← E_tree ∪ {E_tree}
15         V_tree ← V_tree ∪ {x_tree}
16         insertToQ(H_rm, Q, x_tree, δ_hr)
17         if goalReached(X_goal, x_tree) then
18             return Path(G_tree, x_start, x_tree)
19 return NULL
```

Algorithm 4: $\texttt{insertToQ}(\mathbf{H}_{rm}, \mathbf{Q}, x_{tree}, \delta_{hr})$.

```
1  v_clst ← NearestNeighbor(H_rm, x_tree)
   // d_M is the distance metric
2  δ ← d_M(x_tree, v_clst)
3  if δ ≤ δ_hr then
4      Q.insert(x_tree, δ + h(v_clst))
```

When the base station receives a replan request from an agent, it updates the roadmap using two steps. First, it adds new valid edges from the current robot locations $\mathbf{S}_{cur}$ to the closest valid roadmap nodes. Second, it marks as "invalid" all nodes and edges that intersect with any new obstacles $\mathbf{O}_{new}$ detected by the replan requesting agent. The base station next updates the cost-to-go values in the updated roadmap to all unvisited targets using the function `getHeuristics` (line 7). This function runs Dijkstra's algorithm [22] to compute the cost-to-go values once for each unvisited target. It outputs $\mathbf{H} = (\mathbf{H}_1, \ldots \mathbf{H}_j, \ldots \mathbf{H}_m)$ where $\mathbf{H}_j$ is a tuple list with entry $(v_{rm_k}, h_j(v_{rm_k}))$, where $h_j(v_{rm_k})$ is the cost-to-go value i.e. the shortest distance between roadmap node v_{rm_k} and unvisited target v_{rm_j} in $\mathcal{G}_{rm}$. Since Dijkstra's algorithm requires a connected graph, we assume that we sample enough nodes and create edges at the mission start to maintain a connected roadmap till the mission end, even though we invalidate some nodes and edges as new obstacles are found. We believe this is a valid assumption in our work as the new obstacles tend to be small relative to the size of the navigable regions. However, if we detected a disconnected path to a node in our experiments, we set its heuristic to a large invalid value. Alternatively, we refer the reader to the work by Kallman et al. [21] of using online calculated RRTs to reconnect disconnected nodes.

Algorithm 5: $\texttt{Exploit}(\mathbf{H}_{rm}, \mathcal{U}, x_{pop}, \delta_{hr})$.

```
   // Heuristic edge-generation
1  V_{rm_near} ← { v ∈ H_rm.Nodes() | d_M(x_pop, v) ≤ δ_hr }
2  v_best ← argmin_{v ∈ V_{rm_near}} (d_M(x_pop, v) + h(v))
3  E_new ← BestInputProp(x_pop, v_best, U)
4  return E_new
```

Algorithm 6: $\texttt{Explore}(\mathcal{G}_{tree}, \mathcal{U})$.

```
   // Random edge-generation
1  x_rand ← RandomState()
2  x_near ← NearestNeighbor(x_rand, G_tree)
3  E_new ← MonteCarloProp(x_near, U)
4  return E_new
```

We do not change the central roadmap when a robot detects another robot for several reasons. First, it helps limit the communication and computational cost necessary for updating the roadmap. Second, we assume that robots can accurately detect each other for collision avoidance and build local motion plans to avoid each other. Finally, the central roadmap is used as a focusing heuristic for only a percentage of the local roadmap extensions (other extensions use random sampling).

Finally, the base station generates the robot-target assignment (line 8) utilizing the Hungarian algorithm [23] with the calculated heuristic as the cost. It then sends each agent i, its assigned target j, and the corresponding heuristic list $\mathbf{H}_j$ (line 9). Lastly, the base station informs all agents of the newly detected obstacles (line 11).

B. Framework Component That Runs on an Agent

The agent (Algorithm 2) waits (line 2) until it receives the target assignment (line 3) and heuristic node list (line 4) from the base station. It quickly generates a feasible motion plan (line 6) while leveraging the roadmap heuristics without solving the two-point BVP. Once the robot starts moving along its planned path, it periodically checks if it has reached the assigned target (line 10), or if there is an impending collision with an unknown obstacle (line 14) or other robots (line 20). When the robot encounters any of these three conditions, it executes a safety trajectory [24] to bring itself to rest (lines 11, 17, 21) and handles the corresponding condition. In our work, we define a safety trajectory as a collision-free trajectory with the added constraint that its end velocities are zero. Bringing the robot to rest ensures we have enough time to replan and generate a new path to the assigned target.

In the following subsections, we describe feasible motion planning, target-reached condition, and collision avoidance.

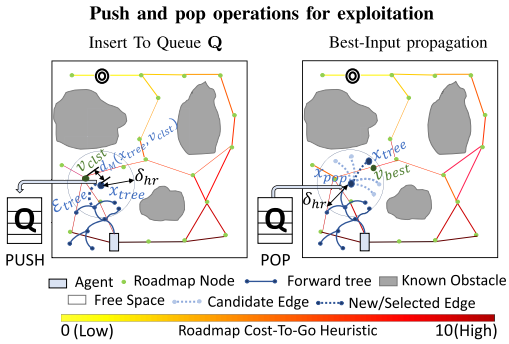

Fig. 3. The left figure shows a search tree node x_{tree} pushed to $\mathbf{Q}$ (Algorithm 4). The right figure shows the same forward tree node popped out from $\mathbf{Q}$ (designated as x_{pop}) and used in exploit step.

C. Heuristic Guided Feasible Sampling-Based Motion Planning

Numerous prior works have been proposed for heuristic guided sampling-based motion planning to improve planning time [4], [25], [26], [27]. Our proposed Algorithm 2 is general enough that any previous work that exploits the roadmap cost-to-go heuristics without solving the two-point BVP [4], [26] can be used in line 6 to produce fast feasible plans. In our experiments, we propose and use a simple roadmap-heuristic guided planner called Heuristic Informed Rapidly-exploring Random Tree (HIRRT) (Algorithm 3), which we briefly outline next. HIRRT uses similar ideas from [4], [26], [27] for heuristic-guided propagation.

Inputs: The inputs to HIRRT (Algorithm 3) are the start node x_{start}, goal region $\mathcal{X}_{goal}$, roadmap heuristic tuple list $\mathbf{H}_{rm}$, the control input set $\mathcal{U}$, and two user-defined input parameters δ_{hr} and q. The parameter δ_{hr} defines the threshold distance within which nodes from the roadmap have a focusing effect on the tree. The parameter q defines the balance of exploration and exploitation strategies performed during search. We use a priority queue $\mathbf{Q}$ to maintain a list of candidate tree nodes for fast expansion (line 2).

Tree expansion: We use a combination of exploitation (line 9) and exploration (line 11) strategies to grow the tree $\mathcal{G}_{tree}$. The exploitation strategy uses the roadmap's heuristic for focused expansion (Fig. 3), whereas the exploration strategy uses random exploration. q decides whether exploitation (Algorithm 5) or exploration (Algorithm 6) is performed in the current iteration by comparing its value against a uniform randomly generated value c_{rand} (line 4). In our experiments, we choose $q > 0.5$ to prioritize exploitation to produce fast, feasible paths. However, a q high value can cause inadequate exploration, potentially causing the search to get stuck near obstacles. We refer the reader to algorithms proposed in [27] for a detailed description of explore and exploit steps.

D. Handling Target-Reached Condition

We assume the target is reached when the robot enters the goal region ($x_i \in \mathcal{X}_{g_j}$). In most cases, the velocities are non-zero as the robot enters the goal region, as we do not constrain them during motion planning. Therefore we execute a safety trajectory to bring the robot to rest (line 11 in Algorithm 2) and then request the base station to replan and reassign targets (line 13 in Algorithm 2). The replanning ensures that all targets are visited as quickly as possible. With the new assignment, a robot previously assigned an unvisited target may not be assigned the same target anymore or given no target.

E. Robot-Obstacle and Robot-Robot Collision Avoidance

The robot monitors for impending collision with new obstacles (line 14 in Algorithm 2) and/or other robots (line 20 in Algorithm 2) as it moves along its planned path. The assumption (Section III) that the sensor radius is at least $2*d_{stop}$ implies that the robot has enough time to execute a safety trajectory and bring it to rest without colliding with obstacles and other previously detected robots. Although this assumption decreases the probability of collision, it does not guarantee collision avoidance if the robot detects a new robot while executing the safety trajectory. This situation can be fixed by continuously monitoring for other robots while executing the safety trajectory.

Additionally, for collision avoidance with other robots, we need to ensure they do not collide when they start moving again. Any previously proposed methods [28] can be used to handle this case. However, our experiments use a simple approach similar to message collision avoidance in the Ethernet CSMA-CA protocol [29]. After an agent comes to rest, it waits for a random amount of time before replanning. Then, if it detects that another agent is already moving, it again waits for a random amount of time before it starts planning again. This way, a robot starts moving again only when it does not detect another moving robot within its sensor range. We choose to wait a random time rather than perform a continuous check to reduce deadlocks between robots and conserve power.

Note that a robot requests a global replan with the base station when encountering a new obstacle. However, it only performs a local replan to the assigned target after avoiding another robot. Finally, the design choice of each robot planning individually causes our multi-robot framework to sacrifice probabilistic completeness [1] for our mission constraints of low computational resources and limited communication. We note that this is a well-known trade-off [30] that is shared by any multi-robot motion planning algorithm that solves the centralized planning problem in the joint state space of the entire team, which becomes computationally intractable for mid to large multi-robot systems.

VI. Experiments

We outline our simulation and hardware setups in Sections VI-A and VI-B, respectively. In both setups, we discretize the workspace and create an occupancy grid to help maintain and invalidate edges of the dynamic costmap [21]. The grid also helps in discretizing obstacles into obstacle cells. We collision-check an edge by checking if any edge part lies in the obstacle cells. The cell direct indexing makes the collision check very fast. Finally, to simplify robot-robot collision avoidance, we exchange the robot's position through the base station but note that any robot detection methods [31] can be used.

A. Simulation Experiments

We use a ROS-based simulation environment of size 25 m $\times$ 25 m of resolution 0.5 m that resembles a planetary surface with known and unknown obstacles. We create three scenarios (Fig. 4), each with a different number and layout of targets and known and unknown obstacles to test different lunar environments. Scenarios 1, 2, and 3 have the number of targets $m = 6, 7,$

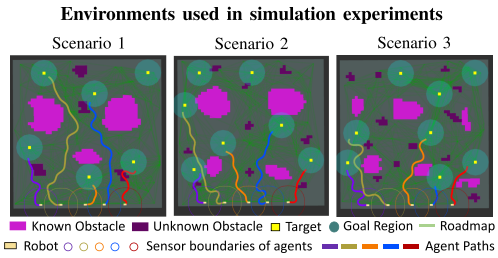

Fig. 4. The pink pixels represent obstacles known prior to simulation onset. The dark violet pixels represent new (smaller) obstacles not known before. The agents' initial paths are shown in different colors.

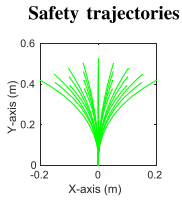

Fig. 5. Potential safety trajectories for a Treaded robot for different values of a_L and a_r with initial $v_L = v_R = 1.0$ in positive y direction.

and 8, respectively. We use $n = 5$ robots in our experiments that are modeled as rectangles with length 0.5 m and breadth 0.3 m.

We use total planning time and mission time as the performance metrics. The total planning time is the sum of the time taken to run the motion planning algorithm on all robots and for all replans. The mission time is the time taken to visit all targets (end of mission) and includes total planning time and traverse time. We do not add the time taken for roadmap construction in the performance metrics as it is calculated offline. However, we include the time taken for the subsequent manipulation of the roadmap in the mission time. We test our algorithm on two different 5-D dynamical systems (Car-Like and Treaded) that are considered viable candidates for the dynamics of planetary rovers.

1) Car-Like: This vehicle [6] has a state (x, y, θ, v, ϕ) where x, y are Cartesian coordinates, θ is the orientation, v is the linear velocity and ϕ is the steering angle. The control space is 2-D, (a, ω), where a is the acceleration and ω is the steering rate. Its dynamical equations are $\dot{x} = v\cos(\theta)\cos(\phi)$, $\dot{y} = v\sin(\theta)\cos(\phi)$, $\dot{\theta} = \frac{v\sin(\phi)}{L}$, $\dot{v} = a$ and $\dot{\phi} = \omega$ where L is the distance between the back and front wheels.

2) Treaded: This vehicle [32] has the state (x, y, ϕ, v_L, v_R) where x, y are Cartesian positions, ϕ is the angle of vehicle from horizontal, and v_L and v_R are the velocities of left and right wheels respectively. The control space is 2-D (a_L, a_R) which corresponds to accelerations of the left and right wheels. Its dynamics is represented by $\dot{x} = v_x\cos(\phi) - v_y\sin(\phi)$, $\dot{y} = v_x\sin(\phi) - v_y\cos(\phi)$, $\dot{\theta} = w_z$, $\dot{v}_L = a_L$, $\dot{v}_R = a_R$, where $v_x = k_1 v_L + k_2 v_R$, $v_y = k_3(v_L - v_R)$ and $\omega_z = k_4(v_L - v_R)$. k_1, k_2 and k_3 are constants that depend on the vehicle's left and right track's instantaneous center of rotation [32]. In our experiments, we choose $k_1 = k_2 = 0.2$, $k_3 = 1.0$ and $k_4 = 1.66$.

We use a Proportional-Derivative (PD) controller to generate safety trajectories and bring the robot to rest. We execute a potential trajectory only if it passes a collision check. We change initial ω for car-like robots and initial (a_L, a_R) for treaded robots (Fig. 5) to get different safety trajectories. We set the subsequent control inputs (a, ω) as $a = -K_{p1}v - K_{d1}a_{prv}$ and $\omega = -K_{p2}\phi - K_{d2}\omega_{prv}$ for car-like vehicle and (a_L, a_R) as $a_L = -K_{p3}v_L - K_{d3}a_{L_{prv}}$ and $a_R = -K_{p4}v_R - K_{d4}a_{R_{prv}}$ for the treaded vehicle where $K_{pi} = 10$ and $K_{di} = 0.5$ are the PD gains, a_{prv}, $a_{L_{prv}}$ (left), $a_{R_{prv}}$ (right) are the previous accelerations and ω_{prv} is the previous angular velocity.

In our experiments, we compare our proposed framework with the HIRRT planner against planners that use different heuristics and show the advantage of deriving heuristics from a centralized dynamic roadmap. For comparison, we replace HIRRT with RRT [33] (5% goal bias heuristic), GBRRT [27] (reverse tree heuristic with dynamics), GABRRT [27] (reverse tree heuristic with no dynamics), and LE [4] (also uses roadmap heuristic) in line 6 of Algorithm 2. As RRT, GBRRT, and GABRRT do not employ roadmap heuristics, we use the euclidean distance between a robot's current position and a target as the cost function in the Hungarian algorithm (line 8) in Algorithm 1. We append the MR (Multi-Robot) to each comparison algorithm to indicate that they are being used within the multi-robot planner framework (Algorithms 1, 2). We include two variants of MR-HIRRT in the comparison. The first is MR-HIRRT-P, which uses PRM, and the second is HIRRT-D, which uses a deterministic roadmap built with nodes on a workspace grid of resolution 0.5 m. We construct roadmaps on the systems' 2-D (x, y) lower-dimension subspace with the initial number of PRM nodes, $N = 2500$. We set HIRRT parameters $\delta_{hr} = 4.5$ and $q = 0.85$. We run 50 trials of each algorithm, with each trial having a different random seed. We use Runge-Kutta Order 4 (RK-4) for performing numerical integration. We run the experiments on a system with Intel i7-7700 4-core CPU with 32 GB RAM.

Next, we provide an analysis (Fig. 8) of the initial number of PRM nodes N vs. total planning time for both car-like and treaded dynamics for Scenario 2. We vary the initial sampled nodes from 64 to 4096 in increments of powers of 2.

Finally, we provide a scalability experiment analysis (Fig. 10) for the 'car-like' robot with n varying from 2 to 8 and m varying from 5 to 25 in increments of 5 in an environment of size 50 m $\times$ 50 m. We run 25 trials for each robot-target datapoint reporting the total planning time and mission time.

B. Hardware Implementation

We run our planner on prototype rovers at the Lunar Mini-yard at NASA JPL. The mini yard is cluttered with rocks and craters to resemble the lunar surface. Each rover has a snapdragon 821 processor (4-core, 1.6 GHz) for processing and a stereo camera for obstacle detection and visual odometry (VO). The robots run ORB-SLAM2 [34] for mapping and localization. We use a LIDAR scanner to scan and build a prior environment map. New obstacles are then placed in the environment to resemble obstacles not detected through a prior map. We use an i7-8750H 6-core CPU with 16 GB RAM as the base station that maintains the centralized dynamic roadmap. The base station communicates with the agents using 802.11n WiFi. We overlay a traversability costmap [35] on top of the occupancy grid of resolution 0.3 m to help with motion planning. The input to the traversability mapping module is the robot-centric elevation map output by the elevation mapping module [35] which takes inputs from the stereo camera. We use a trajectory library (Fig. 6) to emulate not solving two-point BVP (i.e. constraining motion of rovers only in certain directions). We use HIRRT to generate global plans and Timed Elastic Band (TEB) [36] planner as the

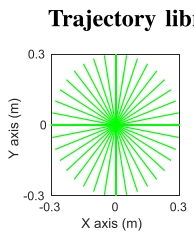

Fig. 6. Trajectory library for hardware experiments to simulate no solution to two-point BVP. For sideways motion, the rover can turn in place.

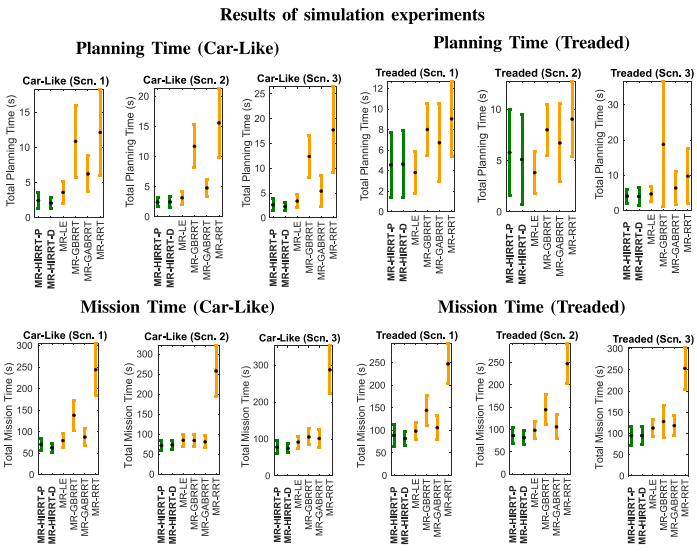

Fig. 7. Top row shows total planning time and bottom row shows mission time (planning time + traverse time) for 50 trials of running each planner for Car-Like (left) and Treaded (right) dynamics. Our proposed planners are highlighted in green.

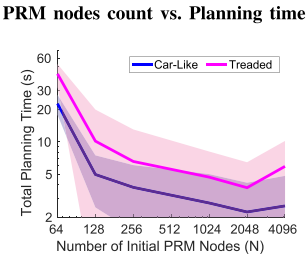

Fig. 8. X and y-axis are in log scale. Each point is mean of 25 trials.

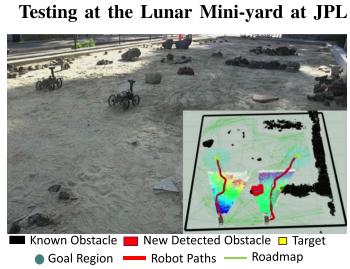

Fig. 9. The ROS rviz inset is displayed in bottom-right. The triangle-shaped regions (colors resemble elevation) represent the sensor footprint of the robot.

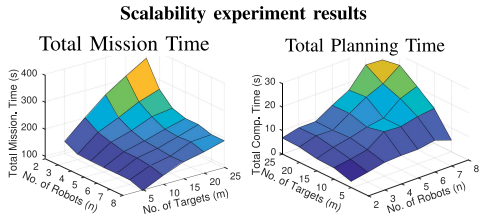

Fig. 10. Scalability experiment results for the 'car-like' robot with each point being a mean of 25 scenarios.

local planner in our experiments. We set $N = 2000$, $\delta_{hr} = 0.27$ m and $q = 0.6$.

VII. Experiment Results

Both MR-HIRRT-P and MR-HIRRT-D (Fig. 7) perform the best in the planning time metric compared to other planners in the majority of the scenarios. This result is because of the more informed heuristic provided by the centralized roadmap compared to other algorithms. Interestingly, both variants of MR-HIRRT perform the best in all scenarios in the total mission time metric. This can be attributed to shorter planning time and a more focused search toward the targets, which leads to more straightforward paths to the goal. Both variants perform equally well in both metrics with no statistical difference. This shows, at least for the scenarios and dynamical systems we tested, that the results are invariant to the underlying sampling technique type. MR-LE [4], which also uses the roadmap heuristic, is faster than other variants that do not use the roadmap heuristic and performs best for Treaded-Scenarios 1 and 2. RRT performs the worst of all algorithms because it uses the least informed heuristic.

The analysis of the planning time vs. the number of roadmap nodes (Fig. 8) shows that the planning time decreases as the sampled nodes increase because of the improved heuristic obtained with more nodes. However for large number of nodes, the planning time slightly increases. This happens because the computation time involved in processing the nodes offsets the heuristic benefit from them.

The scalability analysis (Fig. 10) shows that the total mission time generally decreases as the number of robot increases. This trend is because as the number of robot increases, more robots co-operatively work to complete the mission faster. However, the total planning time generally grows as the number of robots increases. This is because more agents have to replan whenever an agent reaches a target and because of the higher probability of robot-robot collision avoidance instances.

We begin our hardware experiments (Table I) by testing MR-HIRRT using a single robot ($n = m = 1$) and base station for the cases of all unknown and some prior known obstacles. As expected, we noticed a lot more replans for the case of all unknown obstacles. We next test using multiple robots ($n = m = 2$), and the paths found by them are shown in Fig. 9. Although the

TABLE I
Hardware Experiment Results

Scenario	Planning Time (s)	Mission Time (min)	No. of automatic replans	No. of manual interventions
Single Robot (No prior obstacles)	23.63	14.67	7	0
Single Robot (prior)	0.68	6.95	1	0
Two Robots (prior)	0.57	12.2	0	1

robots could reach their assigned targets, the shadows from the surrounding buildings (top left part in Fig. 9) caused issues with the rover's navigation system requiring manual intervention for one robot. Thus a robust navigation system is necessary to be developed that accounts for illumination variations which is outside the scope of this work. We conclude our experiments due to expired access to the rovers and testing space.

VIII. Conclusion

We present a multi-robot motion planning framework to solve the problem of m dynamical agents visiting n unlabeled targets in a partially known environment for rover missions. A base station maintains a centralized roadmap that dynamically updates when robots detect new obstacles. The base station uses this roadmap to generate each agent's cost-to-go heuristic values and the agent-target assignment. The agents use the calculated heuristic to speed up dynamical motion planning (without solving BVP) towards the assigned target. We conduct simulation and hardware experiments to test our proposed planner. We present an analysis of the overall planning time vs. the initial number of PRM nodes. Finally, we provide a scalability experiment analysis with increases in the number of robots and targets.

Our experiments demonstrate that our framework with the HIRRT planner performs better than other planners for most scenarios in the chosen performance metrics. Thus our work shows the advantage of using a centralized dynamic roadmap to accelerate multi-robot motion planning.

References

[1] S. M. LaValle, *Planning Algorithms*. New York, NY, USA: Cambridge Univ. Press, 2006.

[2] "Cooperative autonomous distributed robotic explorers (cadre)," Accessed: Feb. 10, 2022. [Online]. Available: https://www.nasa.gov/directorates/spacetech/game_changing_development/projects/CADRE

[3] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[4] D. Le and E. Plaku, "Guiding sampling-based tree search for motion planning with dynamics via probabilistic roadmap abstractions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 212–217.

[5] A. Bhatia, M. R. Maly, L. E. Kavraki, and M. Y. Vardi, "Motion planning with complex goals," *IEEE Robot. Automat. Mag.*, vol. 18, no. 3, pp. 55–64, Sep. 2011.

[6] L. Duong and E. Plaku, "Cooperative, dynamics-based, and abstraction-guided multi-robot motion planning," *J. Artif. Intell. Res.*, vol. 63, pp. 361–390, 2018.

[7] D. Le and E. Plaku, "Multi-robot motion planning with dynamics via coordinated sampling-based expansion guided by multi-agent search," *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 1868–1875, Apr. 2019.

[8] P. Švestka and M. H. Overmars, "Coordinated path planning for multiple robots," *Robot. Auton. Syst.*, vol. 23, pp. 125–152, 1998.

[9] G. Wagner, M. Kang, and H. Choset, "Probabilistic path planning for multiple robots with subdimensional expansion," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2012, pp. 2886–2892.

[10] G. Wagner and H. Choset, "M*: A complete multirobot path planning algorithm with optimality bounds," in *Redundancy in Robot Manipulators and Multi-Robot Systems*. Berlin, Germany: Springer, 2013, pp. 167–181.

[11] K. Solovey, O. Salzman, and D. Halperin, "Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning," *Int. J. Robot. Res.*, vol. 35, no. 5, pp. 501–513, 2016.

[12] R. Shome, K. Solovey, A. Dobson, D. Halperin, and K. E. Bekris, "DRRT*: Scalable and informed asymptotically-optimal multi-robot motion planning," *Auton. Robots*, vol. 44, no. 3, pp. 443–467, 2020.

[13] J. P. Van Den Berg and M. H. Overmars, "Prioritized motion planning for multiple robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2005, pp. 430–435.

[14] D. Le and E. Plaku, "Cooperative multi-robot sampling-based motion planning with dynamics," in *Proc. Int. Conf. Automated Plan. Scheduling*, 2017, pp. 513–521.

[15] L. Duong and E. Plaku, "Multi-robot motion planning with unlabeled goals for mobile robots with differential constraints," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 7950–7956.

[16] S. H. Shwail, A. Karim, and S. J. Turner, "Probabilistic multi robot path planning in dynamic environments: A comparison between a* and DFS," *Int. J. Comput. Appl.*, vol. 82, no. 7, pp. 29–34, 2013.

[17] J. Tordesillas and J. P. How, "MADER: Trajectory planner in multiagent and dynamic environments," *IEEE Trans. Robot.*, vol. 38, no. 1, pp. 463–476, Feb. 2022.

[18] H. Zhu, F. M. Claramunt, B. Brito, and J. Alonso-Mora, "Learning interaction-aware trajectory predictions for decentralized multi-robot motion planning in dynamic environments," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 2256–2263, Apr. 2021.

[19] G. Sartoretti et al., "PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning," *IEEE Robot. Automat. Lett.*, vol. 4, no. 3, pp. 2378–2385, Jul. 2019.

[20] M. Damani, Z. Luo, E. Wenzel, and G. Sartoretti, "Primal_2: Pathfinding via reinforcement and imitation multi-agent learning-lifelong," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 2666–2673, Apr. 2021.

[21] M. Kallman and M. Mataric, "Motion planning using dynamic roadmaps," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2004, pp. 4399–4404.

[22] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[23] H. W. Kuhn, "The hungarian method for the assignment problem," *Nav. Res. Logistics Quart.*, vol. 2, no. 1/2, pp. 83–97, 1955.

[24] S. Kousik, S. Vaskov, M. Johnson-Roberson, and R. Vasudevan, "Safe trajectory synthesis for autonomous driving in unforeseen environments," in *Proc. Dyn. Syst. Control Conf. Amer. Soc. Mech. Eng.*, 2017, Art. no. V001T44A005.

[25] M. P. Strub and J. D. Gammell, "Adaptively informed trees (AIT*) and effort informed trees (EIT*): Asymmetric bidirectional sampling-based path planning," *Int. J. Robot. Res.*, vol. 41, no. 4, pp. 390–417, 2022.

[26] Z. Littlefield and K. Bekris, "Efficient and asymptotically optimal kinodynamic motion planning via dominance-informed regions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1–9.

[27] S. Nayak and M. W. Otte, "Bidirectional sampling-based motion planning without two-point boundary value solution," *IEEE Trans. Robot.*, vol. 38, no. 6, pp. 3636–3654, Dec. 2022.

[28] J.-P. Calliess, D. Lyons, and U. D. Hanebeck, "Lazy auctions for multi-robot collision avoidance and motion control under uncertainty," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, 2011, pp. 295–312.

[29] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach Edition*. Boston, MA, USA: Addision-Wesley, 2007.

[30] V. J. Lumelsky and K. Harinarayan, "Decentralized motion planning for multiple mobile robots: The cocktail party model," *Auton. Robots*, vol. 4, no. 1, pp. 121–135, 1997.

[31] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "Collaborative multi-robot localization," in *Mustererkennung*. Berlin, Germany: Springer, 1999, pp. 15–26.

[32] J. Pentzer, S. Brennan, and K. Reichard, "Model-based prediction of skid-steer robot kinematics using online estimation of track instantaneous centers of rotation," *J. Field Robot.*, vol. 31, no. 3, pp. 455–476, 2014.

[33] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, 2001.

[34] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[35] M. Wermelinger, P. Fankhauser, R. Diethelm, P. Krüsi, R. Siegwart, and M. Hutter, "Navigation planning for legged robots in challenging terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2016, pp. 1184–1189.

[36] C. Rosmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Efficient trajectory optimization using a sparse model," in *Proc. IEEE Eur. Conf. Mobile Robots*, 2013, pp. 138–143.