Adaptive Coverage Path Planning for Efficient Exploration of Unknown Environments

Amanda Bouman[1], Joshua Ott[2], Sung-Kyun Kim[3], Kenny Chen[4],
Mykel J. Kochenderfer[2], Brett Lopez[5], Ali-akbar Agha-mohammadi[3], Joel Burdick[1]

Abstract— We present a method for solving the coverage problem with the objective of autonomously exploring an unknown environment under mission time constraints. Here, the robot is tasked with planning a path over a horizon such that the accumulated area swept out by its sensor footprint is maximized. Because this problem exhibits a *diminishing returns* property known as submodularity, we choose to formulate it as a tree-based sequential decision making process. This formulation allows us to evaluate the effects of the robot's actions on future world coverage states, while simultaneously accounting for traversability risk and the dynamic constraints of the robot. To quickly find near-optimal solutions, we propose an effective approximation to the coverage sensor model which adapts to the local environment. Our method was extensively tested across various complex environments and served as the local exploration algorithm for a competing entry in the DARPA Subterranean Challenge.

I. INTRODUCTION

Consider a time-limited mission wherein a ground robot must autonomously explore an unknown environment with complex terrain. The robot explores by maximizing the area observed, or *covered*, by a task-specific *coverage sensor*. This sensor may be a thermal camera for detecting thermal signatures, an optical camera for identifying visual clues, or in our case, an omnidirectional range finder for constructing 3D environment maps. As the robot moves, the sensor footprint sweeps the environment, expanding the covered area, or more generally, the task-relevant information about the world. The problem of finding efficient and safe coverage trajectories is computationally complex [1], [2]– one must consider the fact that a robot's observation of the world affects the utility of future observations, while concurrently minimizing traversability risk.

Our proposed method quickly finds non-myopic coverage paths by rolling out future coverage observations using an effective sensor model. Our model is carefully designed to
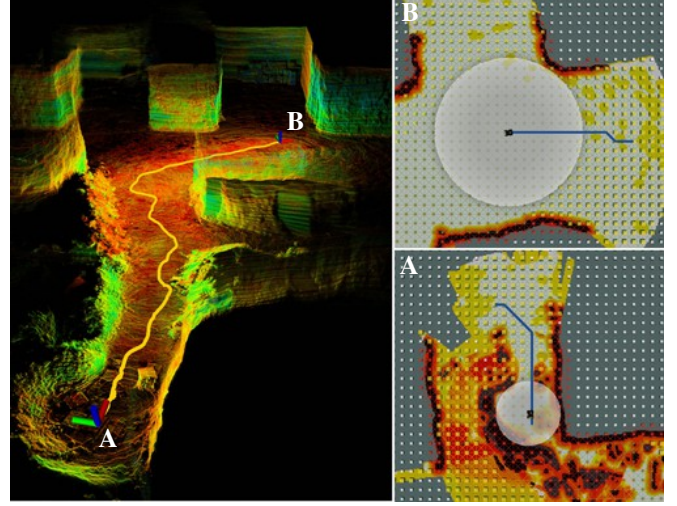
[1]Department of Mechanical and Civil Engineering, California Institute of Technology (e-mail: {abouman, jwb@robotics}.caltech.edu).

[2]Department of Aeronautics and Astronautics, Stanford University (e-mail: {joshuaott, mykel}@stanford.edu).

[3]NASA Jet Propulsion Laboratory, California Institute of Technology (e-mail: {sung.kim, aliahga}@jpl.nasa.gov).

[4]Department of Electrical and Computer Engineering, University of California Los Angeles (e-mail: kennyjchen@ucla.edu).

[5]Department of Mechanical and Aerospace Engineering, University of California Los Angeles (e-mail: btlopez@ucla.edu).

Fig. 1: Adaptive coverage range (translucent circle) and resulting exploratory path (blue) in a locally confined area (A) and a spacious area (B) during Husky's autonomous exploration of a limestone mine in Nicholasville, KY.

replicate critical features of a range finder in a computationally efficient manner. First, the model is probabilistic – coverage probability decreases with increasing ray sparsity along the radial direction. As an effect, the density of coverage is dictated by the local environment geometry, and large topological features in the environment are quickly exposed and mapped. Second, to account for ray-surface interactions that regulate surface visibility, the coverage range, or distance at which a sensor measurement is performed, adapts to the scale of the local environment. This approach obviates the need for expensive ray-tracing operations that make forward rollout algorithms prohibitively slow for a real-time system.

We begin by noting that the coverage task is submodular. Since the robot must understand the effects of its actions on the quality of future coverage measurements, we choose to formulate this problem as a sequential decision process. To find near-optimal trajectories at high replanning rates, we use an online forward rollout search algorithm that plans from the current world-robot state to a travel budget-defined horizon. Our method was evaluated on hardware in various environments, and served as the local planner for team CoSTAR's entry in the Final Circuit of the DARPA SubT Challenge [3].

II. RELATED WORK

The problem of finding the optimal sequence of sensing actions, or viewpoints, in order to maximize some task-

specific information has been extensively studied, both in computer vision and robotics. In the robotics field, the problem of viewpoint selection is commonly motivated by tasks such as surveillance, object inspection, and exploration. While a variety of viewpoint selection algorithms have been proposed, we address those used to solve the exploration problem where policies are constructed in a receding horizon fashion as the robot gathers more sensory information about its environment.

Viewpoint selection algorithms employ a sensor model to determine future sensing locations that maximize scene information. In the context of exploration, these schemes often rely on identification of the boundary between unmapped and mapped space, regions termed *frontiers*, and seek new robot poses that extend the boundary of mapped space [4]. Traditional frontier-based approaches construct one-step lookahead policies that find the next most favorable sensing action, the quality of which is determined by the amount of unmapped area that can be visualized [4], [5]. Underpinning many approaches is the next-best-view planner (NBV) [6], where a rapidly exploring random tree is constructed. Each vertex represents a viewpoint, and the vertex that maximizes a utility function, weighing volumetric gain against path distance, is greedily selected as the next goal [7]. Dang et al. [8] extends this strategy by sampling a set of paths, and then selects the path which maximizes volumetric gain. While computationally efficient, NBV-based planners are greedy and therefore susceptible to local minima, leading to suboptimal decision making. An accumulation of suboptimal local decisions can significantly reduce the amount of sensor information gathered over time.

In order to optimize viewpoint selection over a multi-step horizon, the exploration problem has been framed as a variant of the *art gallery* problem [9]. Here the objective is to find a minimal set of viewpoints that maximizes coverage of an area. A critical feature of this problem is the fact that the marginal benefit of selecting a new viewpoint decreases as the set of already selected viewpoints increases – a property known as *submodularity*. A greedy algorithm has been shown to provide a good approximation of the optimal solution to the submodular function maximization problem [10].

Leveraging the effectiveness of greedy methods for sub-modular maximization, many have adopted a decoupled approach to the exploration problem [11], [2], [12]. First, sensing locations are selected using a greedy algorithm. Then a path through the locations is determined. For instance, in the work of Cao et al. [11], a set of viewpoints is first sampled from a grid-based environment representation. Then viewpoints are selected in order of marginal coverage reward. To account for submodularity, the coverage rewards of the remaining viewpoints in the set are recomputed after each selection. The final ordering of viewpoints is determined by solving the standard *traveling salesman problem*. While a decoupled approach provides a non-myopic solution in a computationally efficient manner, we contend that it can be sensitive to model uncertainty, which we discuss in Section IV-C.
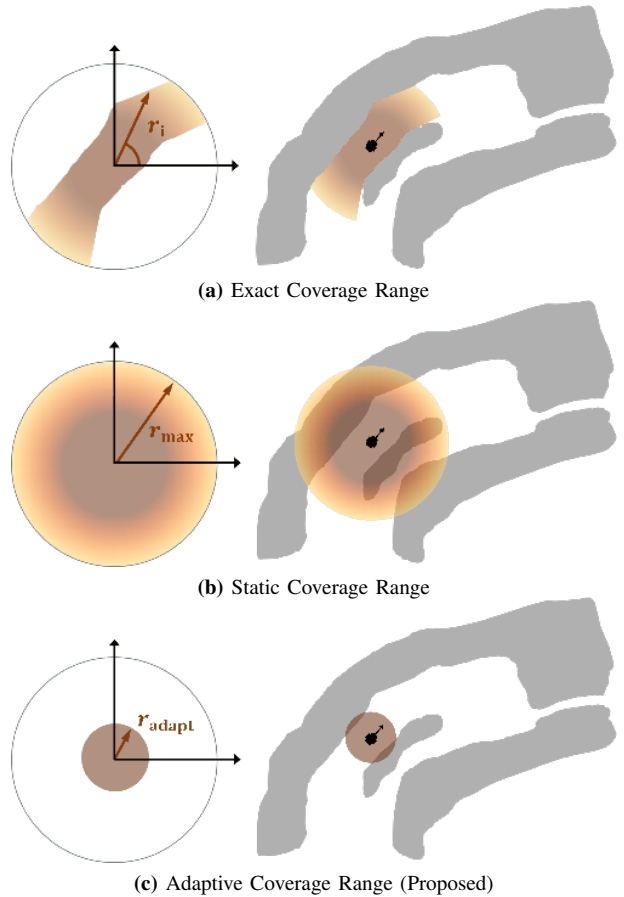

(a) Exact Coverage Range

(b) Static Coverage Range

(c) Adaptive Coverage Range (Proposed)

Fig. 2: Illustrative example of the effect of different coverage sensor models on exploration completeness: "exact" observation where the coverage range is based on ray-tracing (a), approximate observation where the coverage range is static (b), and our proposed approximate coverage sensor model where the range adapts to the local environment. While the exact model provides the best estimate of future coverage, it is computationally expensive and prevents proper investigation of the policy space during MCTS. Alternatively, while the static model is inexpensive, it overestimates the covered area. As a consequence, the passageway below the robot may not be explored since it provides erroneously low coverage reward.

The main contribution of our work is a unified approach to the exploration problem that simultaneously considers environment coverage and robot traversability using a rollout-based search algorithm. The tractability of this approach relies on an approximation of the robot's coverage sensor model, which reduces planning time by adapting to the local environment. We contend that our unified approach is more robust to real-world uncertainty than the widely-adopted decoupled method.

III. PROBLEM DEFINITION

Given a known environment represented by an abstract graph structure $G = (N, E)$, with free and occupied nodes $N_{free} \cup N_{occ} = N$, the coverage objective is to find a sequence of nodes $p = \{n_0, ..., n_{k-1}\} \subseteq N_{free}$ of arbitrary length k such that the number of free nodes within an accumulated coverage sensor footprint F is maximized,

subject to a budget constraint:

$$p^* = \underset{p}{\arg\max} \sum_{n_i \in p} F(n_i), \tag{1}$$
$$\text{subject to} \quad a(p) \le a_{\max},$$

where $a(p)$ is the path action cost, $a_{\max}$ is a user-defined action cost budget, and the sensor footprint F maps each node to a set of "covered" nodes: $F(n_i) = (n_{i_1}, n_{i_2}, .., n_{i_j})$.

Recall that the coverage problem exhibits submodularity; that is, the marginal benefit of appending the path with a node n_2 "close" to $n_1 \in p$ is less than that if $n_1 \notin p$. To account for this *diminishing returns* property, we define marginal coverage as the newly covered area, given all the previously visited nodes:

$$\tilde{F}(n_i) = F(n_i \mid n_0, .., n_{i-1}). \tag{2}$$

Given this definition, we can recast Eq. 1 as a coverage problem with an additive reward structure:

$$p^* = \underset{p}{\arg\max} \sum_{n_i \in p} \tilde{F}(n_i), \tag{3}$$
$$\text{subject to} \quad a(p) \le a_{\max}.$$

We refer to Eq. 3 as our coverage problem for the remainder of the paper.

IV. METHODOLOGY

We model the coverage problem as a discrete-time sequential decision making process where the optimal policy is a sequence of actions chosen to maximize a cumulative coverage reward. To find near-optimal policies in real-time, we employ a rollout-based search algorithm that estimates the value of an action sequence by simulating interactions between the robot and world. During a simulated episode, or rollout, the robot and world states evolve *together* – the robot executes an action and makes a coverage measurement of its environment, Eq. (2), which yields a subsequent robot-world state and reward. Thus, rollouts provide a method of solving the inherently submodular coverage problem in a unified manner, i.e. a policy is evaluated on both the accumulated marginal coverage reward and the path cost.

We introduce our world representation (Section IV-A), and then model our coverage problem as a Markov decision process (Section IV-B). To solve this problem in real-time on a computationally-constrained robot, we propose an effective approximation to the coverage sensor model, which significantly reduces rollout computation. As a result, we are able to construct high-quality coverage paths at a high planning rate (Section IV-C).

A. World Representation

We represent the local environment around the robot by an information-rich graph structure called the Information Roadmap (IRM) [1], as shown in Fig. 3. The IRM is a fixed-size lattice graph $G = (N, E)$ with nodes N and edges E. Nodes represent discrete areas in space, and edges represent actions. We store two type of information in the IRM: *(i)* the traversability risk of the world with respect
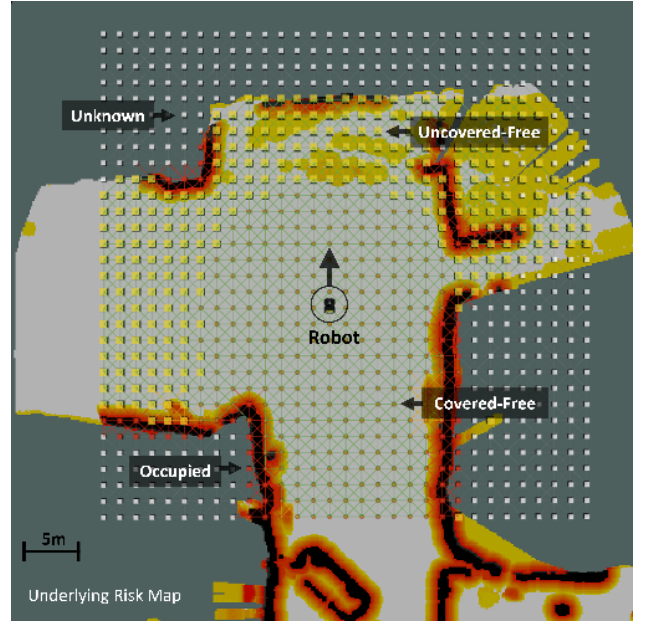

Fig. 3: Information Roadmap (IRM) shown overlaid on the cost map. The IRM contains world coverage and traversability risk information. The goal of the coverage planner is to construct paths on the IRM that convert nodes from uncovered traversable (yellow) to covered traversable (brown). By constructing coverage paths in a receding-horizon fashion, the robot extends the boundaries of explored space.

to the robot's dynamic constraints, and *(ii)* what parts of the environment have been observed, or *covered*, by a task-specific coverage sensor. The robot-centered, rolling window IRM is continuously updated with traversability and coverage information based on incoming sensor data.

To construct G, we uniformly sample nodes $n_i \in N$ in a neighborhood of the robot, and compute the traversability risk and coverage probability distribution over a discrete patch centered at each node, i.e., $p_r(n_i)$ and $p_c(n_i)$, which are stored as node properties. For scalability, we bin node traversability risk probabilities into three groups: *occupied* $p_r(n_i) = 1$, *unknown* $p_r(n_i) = 0.5$, and *free* $p_r(n_i) = 0$. For an edge $e_{ij} \in E$, we compute and store the traversal distance d_{ij} and traversal risk ρ_{ij} between two connected nodes.

B. Markov Decision Process

A Markov decision process (MDP) is described as a tuple $\langle \mathbb{S}, \mathbb{A}, T, R \rangle$, where $\mathbb{S}$ is the set of joint robot-and-world states, and $\mathbb{A}$ is the set of robot actions. The motion model $T(s, a, s') = p(s' \mid s, a)$ defines the probability of being in state s' after taking action a in state s, and the reward function $R(s, a)$ returns the utility for executing action a in state s. The objective is to find a mapping from states to actions, i.e. the policy π, that maximizes the expected sum of future reward.

State: The robot-world state is defined as $s = (q, W)$, where q is the robot state and W is the world state. We define q and W in terms of the IRM. The robot state $q = (n_q, \mu)$, where n_q is the node closest to the robot's current location,

and μ is the robot's heading direction, defined with respect to the lattice geometry. The world state is $W = G$, where G is the IRM containing traversability risk and coverage world state estimates.

Action: We define an action a as the controlled robot traversal from node $n_i \in N$ to neighboring node $n_j \in N$, along an edge $e_{ij} \in E$. A node is directly connected to its eight neighbors, discretizing the valid action space for a single state into movement along the four cardinal/non-diagonal (N, E, S, W) and four intercardinal/diagonal (NE, SE, SW, NW) directions. We denote actions along the cardinal and intercardinal directions by $a_{\sqrt{2}}$ and a_1, respectively.

Robot Dynamics: We approximate the robot motion model $T(q, a, q')$ as deterministic. Given an action a directing traversal of edge e_{ij}, the robot will reach node n_j with probability 1. Actions that cause the robot to leave the bounds of G or enter nodes that are unknown or occupied, $p_r(n_i) = 0.5$ or 1, have no effect. Note that while don't explicitly model motion stochasticity, we account for it by planning at a high-rate in a receding-horizon fashion.

Probabilistic Coverage Sensor Model: We model our coverage sensor as an omnidirectional range finder. The robot covers nodes within its line-of-sight, computed using ray-tracing techniques on the traversal risk map $\{p_r(n_i)\}$ in combination with sensor range constraints. To account for increasing ray sparsity in the radial direction, we compute the coverage probability for a node as a function of the robot-to-node distance. Given the robot node n_q, a node n_i is covered with probability $P_{\text{cov}}(n_i|\ n_q)$. We heuristically model the coverage probability P_{cov} as an S-shaped logistic function:

$$P_{\text{cov}}(n_i|\ n_q) = \frac{1}{1 + e^{k\,(r_i - r_0)}}, \quad (4)$$

where r_i is the euclidean distance between the robot node n_q and node n_i, and constants r_0 and k are the sigmoid's midpoint and steepness, respectively. The coverage probability distribution over the radial distance from the center of the sensor in shown in Fig. 2.

World Transition Model: We approximate the world transition function $T(W, a, W')$ as deterministic. Function CoverageUpdate in Alg. 1 presents the process for updating the world coverage state based upon the the probabilistic coverage sensor model in Eq. (4). When integrating new sensor measurements, we assume independence and compute the maximum of the old and new coverage probability (Alg. 1-line 5). This yields an optimistic estimate of coverage.

Reward Function: We now redefine our marginal coverage from Eq. (2) to be the uncertainty reduction in the world coverage state induced by an action a:

$$I(s, a) = \sum_{n_i \in N} \beta\Big(p_c(n_i\,|\,a) - p_c(n_i)\Big), \quad (5)$$

where β controls the reward received from covering a node based on its occupancy status. Due to its sparsity, the IRM sometimes fails to identify nodes as occupied in high risk regions. For instance, in Fig. 3, the environment boundary is

---

Algorithm 1 World Coverage Update

Function CoverageUpdate
Input: robot node n_q
       world state G
       maximum sensor range $r_{\max}$
1: **for** all angles θ_k of range finder **do**
2:   **for** all nodes n_i along ray from n_q in direction θ_k **do**
3:     Compute robot-to-node euclidean distance r_i
4:     **if** $p_r(n_i) < \rho_{\max}$ and $r_i < r_{\max}$ **then**
5:       $p_c(n_i)' \leftarrow \max\big[p_c(n_i), P_{\text{cov}}(n_i|\ n_q)\big]$ ▷ Eq. (4)
6:     **else**
7:       **break**
8: **return** $\{p_c(n_i)'\}$

(a) Coverage Probability (Continuous)   **(b)** Coverage Probability (Discretized)

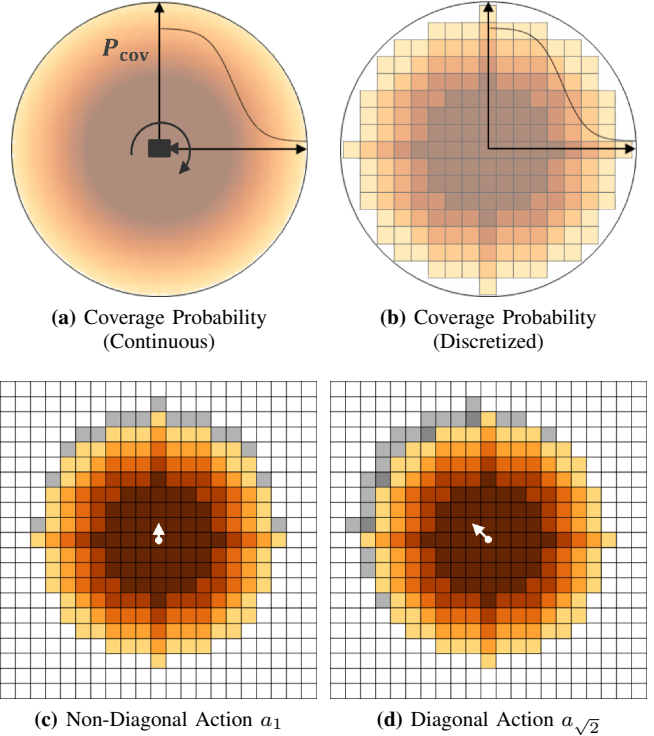**(c)** Non-Diagonal Action a_1 **(d)** Diagonal Action $a_{\sqrt{2}}$

Fig. 4: Our coverage sensor model, based on Eq. (4), displayed over continuous space (a), and over the discretized lattice graph world representation (b). The diffused color map mimics the coverage probability curve– darker shades indicate higher coverage probabilities. The marginal coverage after a non-diagonal action (c) and diagonal action (d) is represented by the shaded gray cells. Note that the ratio of marginal coverage to distance traveled over the lattice is not equivalent for non-diagonal and diagonal actions: $I(s^o, a_{\sqrt{2}})/d_{ij} \neq I(s^o, a_1)/d_{ij}$, where s^o indicates a risk-free world. We address this discrepancy with Eq. (7).

not fully represented by occupied nodes. To stay robust to this unreliable world model, we define the value of β to be larger for nodes of known occupancy (occupied, uncovered-free, and covered-free), when compared to the value of β for unknown nodes. As a result, the constructed coverage paths are more likely to stay within the traversable space of the environment.

The reward function is defined as a weighted sum of marginal coverage and action penalties:

$$R(s, a) = k_I\, I(s, a) - \big[k_d\, d_{ij} + k_\rho\, \rho_{ij} + k_\mu\, \Delta_\mu\big], \quad (6)$$

where d_{ij} is the traversal distance, ρ_{ij} traversal risk, and Δ_μ is the cost of rotation due to the robot's non-holonomic constraints. Constants k_I, k_d, k_ρ, and k_μ weigh the importance of coverage, traversal distance, risk, and motion primitive history on the total reward.

Given a coverage sensor with a circular field-of-view, the uncovered area after a diagonal and non-diagonal action should scale equivalently with distance traveled. However, since Eq. (5) is evaluated over a discretized space G, the ratio of marginal coverage to distance traveled is not equivalent for all actions on the lattice, as illustrated in Fig. 4. Given this marginal coverage discrepancy between actions, we define k_d as a function of coverage parameters in order to ensure non-diagonal (a_1) and diagonal actions ($a_{\sqrt{2}}$) are equally rewarding; that is, $R(s, a_1) = R(s, a_{\sqrt{2}})$ for the same ρ_{ij} and Δ_μ. If w is the width of a grid cell in G, then we define k_d as:

$$k_d = \frac{k_I}{w} \cdot \frac{I(s^o, a_{\sqrt{2}}) - I(s^o, a_1)}{(1 - \sqrt{2})} \qquad (7)$$

where state s^o denotes a risk-free world where the only covered region is aligned with the robot's current sensor footprint.

Optimal Policy: It is fundamentally infeasible to solve an unknown environment coverage problem over an infinite horizon since information about the world is incomplete, and often inaccurate, at runtime. Instead, in such domains, a Receding Horizon Planning (RHP) scheme has been widely adopted as the state-of-the-art [6]. The optimal policy with RHP is:

$$\pi^*_{t:t+T}(s) = \underset{\pi \in \Pi_{t:t+T}}{\operatorname{argmax}} \sum_{t'=t}^{t+T} \gamma^{t'-t} R(s_{t'}, \pi(s_{t'})), \qquad (8)$$

where T is a finite planning horizon for a planning episode at time t. Given the policy from the last planning episode, only a part of the optimal policy, $\pi^*_{t:t+\Delta t}$ for $\Delta t \in (0, T]$, will be executed at runtime. A new planning episode will start at time $t + \Delta t$ with updated robot-world state.

C. Online Planning

We now discuss our proposed online coverage planner algorithm, which runs in real-time on hardware. Alg. 2 presents the major components of the planner.

Search Algorithm: In order to solve Eq. (8), we use Monte Carlo tree search (MCTS) [13]. Refer to Function MCTS in Alg. 2. During every planning episode, a lookahead tree, rooted in an initial robot-world state, is iteratively constructed by simulating action sequences using a random rollout policy $\pi_{rollout}$. During a single iteration, rollouts and tree expansion stop when a predefined depth, or our path budget, is reached. Given a state s and action a, a generative model $\mathcal{G}$ (i.e. the black box simulator of the MDP) provides a sample successor state s' and reward r. Since we do not have access to the ground truth state of the environment, our generative model is an estimate based on the most recent robot sensor measurements used to construct

---

Algorithm 2 Coverage Planner

Function CoveragePlan
repeat
  Obtain: state $s = (n_q, \mu, G)$
        pointcloud scan $\{z_i\}$
  #1 Generate Coverage Mask
  Compute adaptive coverage range r_{adapt} in Eq. (10)
  $\{m_i\} \leftarrow$ CoverageUpdate$(n_q, O, r_{\text{adapt}})$ ▷ Alg. 1
      ▷ where $O \Rightarrow p_c(n_i) = p_r(n_i) = 0 \; \forall \; n_i \in N$
  #2 Find Planning Root
  $n_\tau, \mu_\tau \leftarrow$ RootNode$(s, a^-_{1:N})$ ▷ see PLGRIM in [1]
  $s \leftarrow (n_\tau, \mu_\tau, G)$ ▷ update robot state to root parameters
  #3 Plan and Execute
  $T_r \leftarrow$ MCTS$(s, \{m_i\})$
  Extract action sequence $a^*_{1:N}$ from T_r
  #4 Prep for Next Episode
  $a^-_{1:N} \leftarrow a_{1:N}$
until timeout

Function RootNode
Input: state $s = (n_q, \mu, G)$
      previous action sequence $a^-_{1:N}$
Extract path $a^-_{Q:N}$ ▷ n_Q is path node closest to n_q
Initialize path risk ρ_{path} and distance d_{path} to 0
for action e_{ij} in path $a^-_{Q:N}$ **do**
  ρ_{path} += ρ_{ij}/d_{ij}; d_{path} += d_{ij}
  if $\rho_{\text{path}} > \rho_{\max}$ or $d_{\text{path}} > d_{\max}$ **then**
    Assign root node $n_\tau \leftarrow n_i$
    Find root orientation μ_τ ▷ if $n_\tau = n_Q$, then $\mu_\tau \leftarrow \mu$
    return n_τ, μ_τ

Function MCTS
Input: state $s = (n_q, \mu, G)$
      coverage mask $\{m_i\}$
Initialize empty lookahead tree T_r
repeat
  $T_r \leftarrow$ SIMULATE$(s; \mathcal{G})$
    ▷ estimated generative model $\mathcal{G}$ given by
      Simulate$(s, \{m_i\}; \pi_{rollout})$
until timeout
return T_r

Function Simulate
Input: state $s = (n_q, \mu, G)$
      coverage mask $\{m_i\}$
      policy π
$n'_q, \mu' \leftarrow \pi(n_q, \mu)$
$\{p_c(n_i)'\} \leftarrow \{\max[m_i, p_c(n_i)]\}$ ▷ fast coverage update
$r \leftarrow R(s, a)$ ▷ Eq. (6)
return s', r

---

the world representation G. MCTS terminates after reaching a user-defined maximum number of simulations.

Action Sequence Extraction: The action sequence with the highest estimated value is extracted from the lookahead tree (Alg. 2–#3). Then the first N actions from that sequence, $a^*_{1:N}$, is sent to the robot for execution. The number of actions N is defined such that $R(s_i, a_i) > \gamma \; \forall \; i \in \{1 : N\}$, where γ is an empirically selected one-step reward lower bound. This cropping of the action sequence is critical to global exploration performance; it ensures the local coverage path uncovers "enough" area to justify the path travel cost. If $a^*_{1:N}$ is empty, then a global planner takes control and guides the robot to areas with high expected information gain.

Planning Root Update: At the end of every planning episode, $a_{1:N}^*$ is stored and then used to update the root of the lookahead tree during the subsequent episode (Alg. 2 – #2). Our root update approach is based on a receding-horizon policy reconciliation method proposed by [1]. Fig. 6 demonstrates the effectiveness of this root update method.

Adaptive Coverage Range: While MCTS is an anytime algorithm, meaning construction of the tree can terminate at any point and a solution will be recovered, it only converges to the optimal solution with a sufficient number of simulations. Although it may be infeasible to reach the optimal solution given time constraints, estimates of the action values become increasingly more reliable with more simulations, leading to a higher quality coverage path. In order to find quality solutions at high planning rates, a real-time system must find a good balance between the fidelity of a simulation (e.g. how accurately we model the coverage observation) and the number of simulations.

To maximize the number of simulations within a suitable planning time, we propose an approximation of the coverage model that reduces the time complexity of the generative model $\mathcal{G}$. Our approximate world coverage update obviates the need for expensive ray-tracing operations in Alg. 1. First, we estimate the spaciousness r_{spac} of the local environment [14]. Then we adapt the distance at which a range-finder coverage measurement is performed based on r_{spac}. We denote this adaptive coverage distance by r_{adapt}. See Fig. 2 as an example of our adaptive coverage range approach.

Given a range-finder 3D pointcloud scan $\{z_i\}$ where z_i is the point at which a ray intersects an obstacle, we compute spaciousness as:

$$r_{\mathrm{spac}} = f\big(\mathrm{median}\{d(z_i)\}\big), \tag{9}$$

where $d(z_i)$ is the euclidean distance between the range-finder origin and a ray intersection-point z_i, and f is a low-pass filter: $f(x_t) = \alpha_1 f(x_{t-1}) + \alpha_2 x_t$ with constants $\alpha_1 = 0.95$ and $\alpha_2 = 0.05$. The median is robust to outliers in a potentially noisy pointcloud, and gives a notion of the current scale of the local environment around the robot. Then, given r_{spac}, we compute r_{adapt} as

$$r_{\mathrm{adapt}} = \begin{cases} \alpha \cdot r_{\mathrm{spac}}, & \text{if } r_{\mathrm{spac}} \le \frac{r_{\max}}{\alpha} \\ r_{\max}, & \text{otherwise,} \end{cases} \tag{10}$$

where α is an empirically tuned scaling constant, and $r_{\max}$ is our model-defined maximum sensor range. Equipped with r_{adapt}, we generate a probabilistic coverage mask $\{m_i\}$, detailed in Alg. 2 – #1. The mask serves as an input to the generative model Function Simulate in Alg. 2, which updates the world coverage state using inexpensive matrix operations.

Discussion: A decoupled approach to the coverage planning problem leverages a greedy algorithm for non-myopic viewpoint selection, as detailed in Section II. This approximation relies on the fact that selecting more viewpoints never reduces the total coverage reward, since Eq. 3 exhibits monotonicity [10], [15]. While true in theory, this conjecture falters in a real-world exploration domain where the
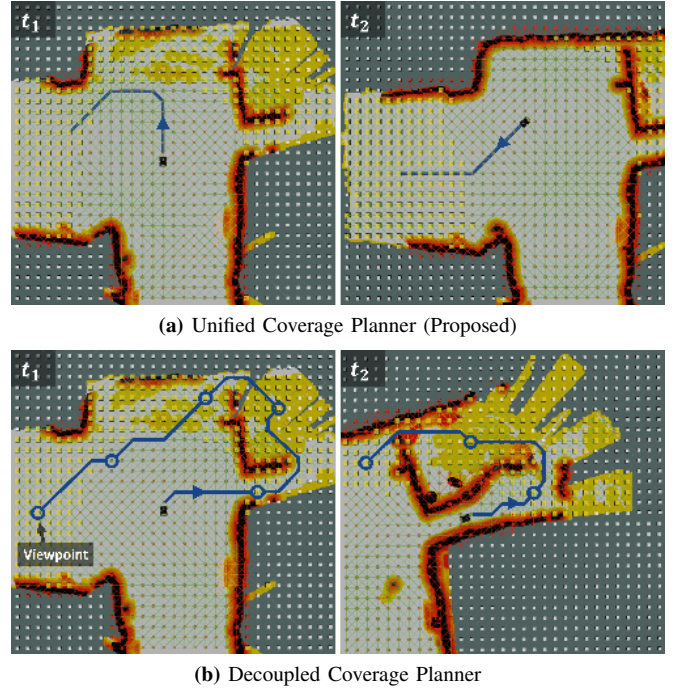

(a) Unified Coverage Planner (Proposed)

(b) Decoupled Coverage Planner

Fig. 5: For two planning episodes at t_1 and t_2 during Husky's autonomous exploration of a real-world mine, we show the coverage path constructed using our proposed unified approach (a) and the commonly-adopted decoupled approach (b) for solving the sub-modular coverage problem. In (a), the robot collects the remaining coverage reward at the end of the passage, before continuing to the large, unexplored passage to the left. In (b) at snapshot t_1, the robot incorrectly detects openings at the end of the passage due to bad sensor measurements and selects a set of viewpoints accordingly. The shortest path through the poorly-selected viewpoints guides the robot through a narrow passage to the right, which is both riskier and less rewarding than the passage to the left.

robot only has partial information about the world. In this setting, the inclusion of risky or low quality viewpoints, i.e., those evaluated using unreliable world estimates, can have adverse effects on the final policy and the robot's ability to collect coverage reward over an exploration mission. More concretely, the policy constructed by a decoupled approach does not consider that: (*i*) the robot may fail during execution of the path, (*ii*) world coverage and traversability estimates become increasingly unreliable with increasing distance from the robot, and (*iii*) the world model (i.e. Local IRM) changes dynamically as the robot uncovers and maps new regions.

In order to address the aforementioned issues, the proposed approach to the coverage planning problem exhibits the following properties that make it suitable for a real-world exploration domain.

1) *Viewpoint Selectiveness*: A policy is evaluated by computing the marginal coverage reward and path cost for each successive action, or viewpoint, in the policy (Eq. 6). Understanding coverage interdependency between successive viewpoints lifts the burden of needing to fully cover the current graph with a single policy – an unproductive and potentially harmful ambition in the presence of uncertainty. As a result, viewpoints that do not provide sufficient coverage utility within
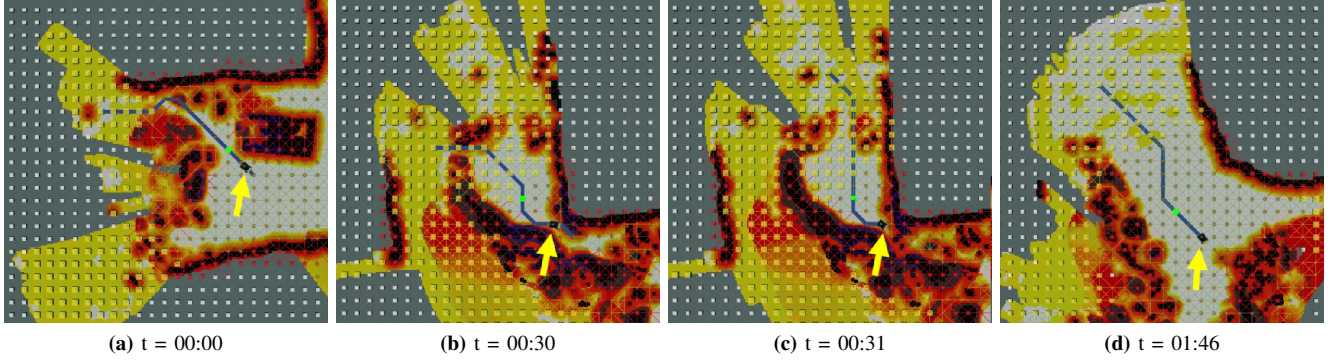
(a) t = 00:00   **(b)** t = 00:30   **(c)** t = 00:31   **(d)** t = 01:46

Fig. 6: Snapshots of robot's navigation through rocks and debris during its exploration of a limestone mine. The coverage path (blue) and the planning root node (green circle) are shown. Note that (b) and (c) are from consecutive planning episodes as the robot turns a corner, receives new sensor information, and updates the world risk state. The policy constructed in (c) is evaluated in (b)'s updated world estimate. The root node location is based on this evaluation (See Function RootNode in Alg. 2).

a time-budget, or jeopardize the robot's safety, can be discounted from the final policy, while still preserving MCTS near-optimality.

2) *Robustness to Uncertainty*: The lookahead tree is rooted at (or very near to) the robot's current location. Hence, MCTS visits nodes close to the robot more frequently, effectively focusing its search time in areas of the environment where world coverage and traversability risk estimates are more reliable. Moreover, due to a discount factor in the problem objective Eq. (8), policies that shift coverage reward earlier in time are more rewarding. By incorporating this near-sighted incentive, the robot accounts for stochasticity in sensing and motion control, as well as the fact that the world model will evolve as undetected areas are exposed.

Fig. 5 compares paths constructed by our approach and a decoupled approach during a real-world exploration mission. Recall that the decoupled approach greedily selects viewpoints in order of highest marginal coverage reward. Therefore, rather than discounting viewpoints far from the robot where world estimates are poor, the decoupled approach actually prioritizes distant points since there is less sensor overlap at these locations with the robot's current field-of-view. The path planner is then "locked into" these viewpoints, and optimistically reasons over this potentially unreliable search space.

V. Experimental Results

In order to evaluate our proposed approach, we performed simulation studies and real-world experiments with a four-wheeled vehicle (Clearpath Robotics Husky robot) and quadruped (Boston Dynamics Spot robot). Both robots are equipped with custom sensing and computing systems [16], [3], [17]. The entire autonomy stack runs in real-time on an Intel Core i7 processor with 32 GB of RAM. The stack relies on a multi-sensor fusion framework, the core of which is 3D point cloud data provided by LiDAR range sensors [18]. During testing, the proposed (or comparative baseline) approach was integrated as the local planner within the hierarchical planning framework PLGRIM [1].

A. Simulation Evaluation

We evaluated the proposed planner against the baseline planner *Ours-LF*: the proposed rollout-based method with a low-fidelity coverage sensor model, i.e. non-probabilistic and static coverage range r_{max}. All tests were performed in a simulated maze environment, as shown in Fig. 7. The maze consists of a large irregular network of large spaces and narrow passages, many of which are connected by sharp bends. This geometry exposes the weaknesses of a rollout-based planner where the coverage sensor model does not effectively approximate the actual range finder sensor. The long-range Ours-LF planner ($r_{max} = 8$m) overestimates the coverage sensor range and, therefore, fails to detect openings at the sharp bends. As a result, large swaths of the environment are not exposed, and the robot terminates exploration early. Alternatively, the short-range Ours-LF planner ($r_{max} = 4$m) performs significantly better since it can expose and explore all narrow passages. However, since it underestimates the coverage sensor range, it finds redundant trajectories in the
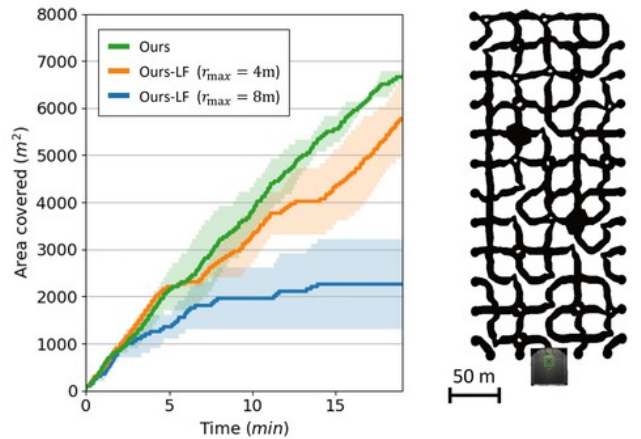

Fig. 7: Results from simulated exploration runs in the simulated maze (shown at top right). We define our coverage metric to be the accumulated area within an 8m radius of the robot. Each curve is the average of 2 runs.

large spaces, which contributes to a slight degradation in performance.

Our proposed solution can handle all settings, since it neither over- or under-estimates the true coverage range in exchange for reducing computation. Moreover, since the model is probabilistic, it inherently adjusts its coverage density to the local environment. As a consequence, when the robot approaches a sharp bend, it travels deep enough to "see" uncovered space around the corner, which is critical to exposing the entire environment.

B. Real-World Evaluation

Our solution was extensively tested on physical robots in real-world environments. In particular, we present results from the exploration of a limestone mine (Figs. 8 and 6) in the Kentucky Underground, Nicholasville, KY.
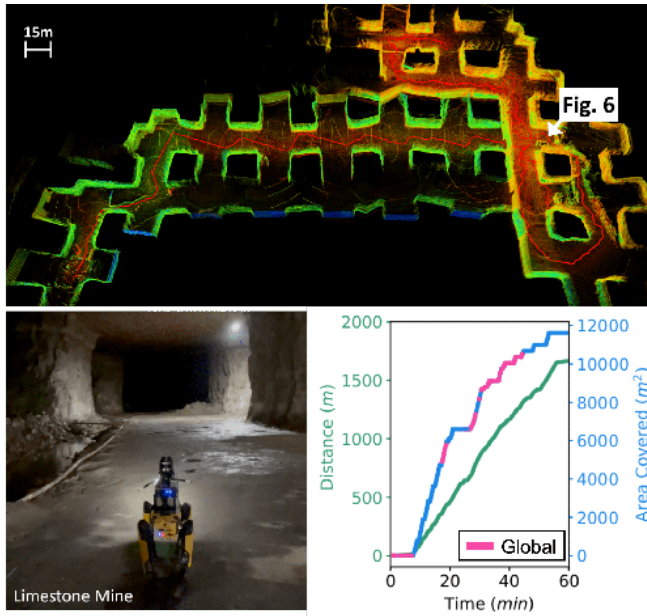

Fig. 8: Proposed coverage planner's navigation of a limestone mine during a 60 min. exploration mission. The coverage paths down the main corridor exhibit a wave-like shape. When the robot encounters a junction, it moves toward the corner in order to maximize coverage of both branches, and then re-aligns with the centerline of the main corridor. In the exploration metric (bottom right), pink denotes the time intervals where the the PLGRIM global planner is directly guiding the robot. The starting location of snapshot (a) in Fig. 6 is indicated.

VI. CONCLUSION

We present an approach for solving the coverage problem for time-constrained autonomous exploration of unknown environments. We solve this problem, which is submodular in nature, using a unified rollout-based search algorithm. This formulation allows us to evaluate the effects of the robot's actions on future world coverage states, while simultaneously accounting for traversability risk and the dynamic constraints of the robot. In order to adequately investigate the search space, we reduce rollout computation using an effective approximation to the coverage sensor model which adapts the coverage range to the local environment. As a result, we can solve the submodular coverage problem in a unified manner, which we contend is more robust to real-world uncertainty than decoupled approaches.

REFERENCES

[1] S.-K. Kim∗, A. Bouman∗, G. Salhotra *et al.*, "PLGRIM: Hierarchical value learning for large-scale exploration in unknown environments," in *International Conference on Automated Planning and Scheduling (ICAPS)*, vol. 31, 2021, pp. 652–662.

[2] L. Heng, A. Gotovos, A. Krause, and M. Pollefeys, "Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1071–1078.

[3] A. Agha-mohammadi and et al., "NeBula: Quest for robotic autonomy in challenging environments; TEAM CoSTAR at the DARPA subterranean challenge," *Journal of Field Robotics*, 2021.

[4] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 1997, pp. 146–151.

[5] H. H. González-Banos and J.-C. Latombe, "Navigation strategies for exploring indoor environments," *International Journal of Robotics Research*, vol. 21, no. 10-11, pp. 829–848, 2002.

[6] A. Bircher, M. Kamel, K. Alexis *et al.*, "Receding horizon "next-best-view" planner for 3D exploration," in *icra*, 2016, pp. 1462–1468.

[7] C. Witting, M. Fehr, R. Bähnemann *et al.*, "History-aware autonomous exploration in confined environments using mavs," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.

[8] T. Dang, M. Tranzatto, S. Khattak *et al.*, "Graph-based subterranean exploration path planning using aerial and legged robots," *Journal of Field Robotics*, vol. 37, no. 8, pp. 1363–1388, 2020.

[9] S. K. Ghosh, *Visibility algorithms in the plane*. Cambridge University Press, 2007.

[10] A. Krause and D. Golovin, "Submodular function maximization." *Tractability*, vol. 3, pp. 71–104, 2014.

[11] C. Cao, H. Zhu, H. Choset, and J. Zhang, "Tare: A hierarchical framework for efficiently exploring complex 3d environments," in *Robotics: Science and Systems Conference (RSS), Virtual*, 2021.

[12] J. Faigl and M. Kulich, "On determination of goal candidates in frontier-based multi-robot exploration," in *2013 European Conference on Mobile Robots*. IEEE, 2013, pp. 210–215.

[13] C. B. Browne, E. Powley, D. Whitehouse *et al.*, "A survey of Monte Carlo Tree Search methods," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 4, no. 1, pp. 1–43, 2012.

[14] K. Chen, B. T. Lopez, A.-a. Agha-mohammadi, and A. Mehta, "Direct lidar odometry: Fast localization with dense point clouds," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2000–2007, 2022.

[15] M. Roberts, D. Dey, A. Truong *et al.*, "Submodular trajectory optimization for aerial 3d scanning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5324–5333.

[16] K. Otsu, S. Tepsuporn, R. Thakker *et al.*, "Supervised autonomy for communication-degraded subterranean exploration by a robot team," in *IEEE Aerospace Conference*, 2020.

[17] A. Bouman∗, M. Ginting∗, N. Alatur∗ *et al.*, "Autonomous Spot: Long-Range Autonomous Exploration of Extreme Environments with Legged Locomotion," in *iros*, 2020.

[18] K. Ebadi, Y. Chang, M. Palieri *et al.*, "LAMP: Large-scale autonomous mapping and positioning for exploration of perceptually-degraded subterranean environments," in *icra*, 2020.