2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

2021 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9077-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICRA48506.2021.9561310

Search-based Planning for Active Sensing in Goal-Directed Coverage Tasks

Tushar Kusnur, Dhruv Mauria Saxena, and Maxim Likhachev

***Abstract*— Path planning for robotic coverage is the task of determining a collision-free robot trajectory that observes all points of interest in an environment. Robots employed for such tasks are often capable of exercising active control over onboard observational sensors during navigation. We address the problem of planning robot and sensor trajectories that maximize information gain in such tasks, where the robot needs to cover points of interest with its sensor footprint. Search-based planners in general guarantee completeness and provable bounds on suboptimality with respect to an underlying graph discretization. However, searching for kinodynamically feasible paths in the joint space of robot and sensor state variables with standard search is computationally expensive. We propose two alternative search-based approaches to this problem. The first solves for robot and sensor trajectories independently in decoupled state spaces while maintaining a history of sensor headings during the search. The second is a two-step approach that first quickly computes a solution in decoupled state spaces and then refines it by searching its local neighborhood in the joint space for a better solution. We evaluate our approaches in simulation with a kinodynamically constrained unmanned aerial vehicle performing coverage over a 2D environment and show their benefits.**

I. Introduction

Path planning for traditional robotic coverage is the task of determining a collision-free robot trajectory that observes all points of interest in a given environment [1]. Numerous real-world tasks including environmental exploration, traffic monitoring, and post-disaster assessment can be cast as robotic coverage problems [2], [3], [4], [5]. Robots employed for such coverage tasks are often equipped with limited-range sensors to observe the environment and can exercise active control over them. An important problem is to plan robot and sensor trajectories that maximize coverage or information gain in these tasks, while also respecting kinodynamic constraints and arriving at a goal location. This is akin to an *informative path planning* problem. Real-time kinodynamic planning is a computationally expensive problem in itself due to the many degrees of freedom in a kinodynamic robot state. Additionally planning trajectories for sensors onboard these robots further increases the computational complexity of this task.

In this paper, we consider the specific problem of planning trajectories for an unmanned aerial vehicle (UAV) and its onboard sensor covering cells of a discrete map—representing a known, deterministic environment—to achieve efficient 2D

The authors are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, USA: {tkusnur, dsaxena, maxim}@cs.cmu.edu. This work was sponsored by Mitsubishi Heavy Industries, Ltd.

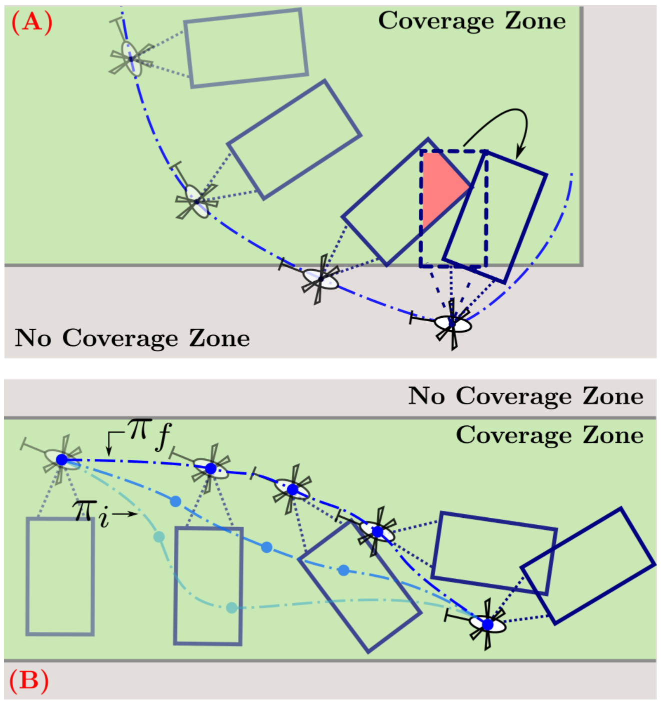


Fig. 1: We present two algorithms for search-based planning for active sensing. (A) SPLASH tries to minimize sensor footprint overlaps along the robot trajectory. For the last state in the figure, it would prefer the solid blue sensor footprint over the dashed blue so as to avoid the overlap denoted by the red area. (B) SPLIT iteratively refines an initial trajectory π_i to maximize the area covered by the sensor to come up with the final trajectory π_f.

area coverage while navigating to an assigned goal. We assume that the UAV flies at a fixed altitude, and that the yaw angle of a pan-only camera onboard the UAV can be controlled, thereby controlling the camera's projected footprint on the ground. In turn, we include the robot's x and y coordinates, heading θ, velocity v, timestamp t, and sensor angle ψ in our state space—making it *at least* a 6-degree-of-freedom (6-DoF) planning problem (we describe in Sec. III how this can amount to planning in more than 6 DoFs). We tackle this problem using a search-based approach, which comes with the advantage of guarantees on solution quality up to the discretization of the graph representing the problem.

However, naïvely applying search-based approaches to such a problem results in high computational cost. A key challenge in planning non-myopic sensor trajectories that maximize coverage is that in general, for a given robot trajectory, the optimal sensor configuration at a given point in the trajectory depends on all previous sensor measurements (the full *sensor history*). One can appreciate this in 2D environments by thinking of sensor footprint overlaps: to compute an optimal sensor configuration at a given location, a planner must take into account all overlaps with previously planned sensor footprints in the plan being considered. However, including the full sensor history in a state makes the search computationally intractable. Our first approach, **S**ensor **Pla**nning with **S**ensor **H**istory (SPLASH), first computes a robot trajectory. It then searches for sensor plans for this fixed robot trajectory while maintaining a *partial* sensor history during the search—this prevents covering areas already sensed before in the trajectory, up to the maintained sensor history horizon. Even for a fixed robot trajectory, the space of sensor trajectories exponentially increases in dimensionality as we account for longer sensor histories (this might not be trivial to see, and so we detail how this occurs in Sec. IV). Our results show that in most planning problems considering 2D sensor footprint overlaps, only a partial history of sensor footprints actually affect computing the next optimal sensor configuration.

The basis of our second approach is the empirical observation that approximate search algorithms like Weighted A* (WA*) overlook better solutions that are actually "close" to the computed solution in the space of solution paths [6], [7]. In our second approach, **S**ensor **P**lanning with **L**ocal **I**terative **T**unneling (SPLIT), we *split* the process into two steps: (1) We first quickly compute suboptimal robot and sensor trajectories in *decoupled* robot and sensor state spaces using SPLASH (initialization). (2) We then use this solution as initialization to a local-search routine that iteratively improves this solution in the *joint* robot and sensor state space until time runs out (refinement). We adapt Iterative Tunneling Search with A* (ITSA*) [7] to our problem for this refinement step. This is detailed in Sec. V.

The illustation in Fig. 1 depicts both of our approaches on a toy example. Our approaches can be contextualized within several related works on sensor management and informative path planning, as we do in Sec. II. In Sec. III, we describe our problem and notation in detail. In sections IV and V, we describe and provide pseudocode for both of our approaches. In Sec. VI, we evaluate our approaches and show their benefits in the context of a previously established planning framework for persistent coverage with multiple UAVs [8], where individual UAVs are tasked with generating collision-free trajectories that maximize coverage while navigating to continuously assigned goals. Note that our contribution lies in planning robot and sensor trajectories for a single UAV navigating to a goal (we do not attempt to solve the problem of coordinating UAV plans for coverage). To the best of our knowledge, no previous work applies search-based planning to this problem.

II. Related Work

Hero and Cochran present an extensive survey on sensor management [9]. Gutpta et al. state general challenges and computational complexity of optimal sensor selection in detail in [10]—this is on similar lines as the computational challenge of maintaining a sensor history (see Fig 4, explained later). In general, optimal coverage has been addressed in various settings including mobile sensors and autonomous robots. Robotic sensing systems have used with both fixed sensors [11], [12] as well as sensors that execute pre-computed patterns [13]. The problem of optimal mobile sensor location with unbounded ranges has been tackled as Voronoi space partitioning in [14]. Many approaches have also been targeted to specific applications, such as active perception work [15], [16]. The measurement control problem, also essentially a sensor scheduling problem, was shown to be solved by tree-search in general [17]. To deal with computational intractability, several greedy solutions have been proposed [10], [18], [19], [20], [21]. Further, Finite-horizon model predictive control provide improvement over myopic techniques but suffer from high run-times in large state spaces and provide no performance guarantees beyond the horizon depth [22], [23]. Arora et al. propose a data-driven approaches to sensor trajectory generation that map calculated features to sensory actions [24].

Several recent works that fall under informative path planning are closely related to our work. Perhaps the most closely related is a recent line of work on active information acquisition, although with *fixed* sensors onboard robots: Atanasov et al. propose a non-greedy, value-iteration based offline solution with applications to gas distribution mapping and target localization [25]. Schlotfeldt et al. then reformulate the problem as a deterministic planning problem and apply A* search with the first consistent heuristic for information acquisition, with applications to active mapping [26]. Kantaros et al. then propose a probabilistically complete and asymptotically optimal sampling-based approach to this problem, along with strategies to bias exploration toward informative regions [27]. Lu et al. propose a potential-function based method for integrated planning and control of robotic sensors deployed to classify multiple targets in an obstacle-populated environment [28].

There are also lines of work that formulate information gathering as an Orienteering Problem. Of particular interest are [29], [30], [31], [32] because of a similarity in their approaches with our initialize-and-refine approach in SPLIT (detailed further in Sec. V). However these approaches focus on computing informative *tours*—unlike our goal-directed setting, and perform local refinement over heading angles constrained by Dubin's-car dynamics—unlike our approach that refines sensor angles that observe the environment.

III. Problem Formulation and Notation

A. Persistent coverage framework

We contextualize and evaluate our approaches within the persistent-coverage framework established in previous work [8]. This is a centralized framework that continuously computes goal locations to which UAVs should fly and kinodynamically feasible, globally deconflicting plans for them to do so, in a *prioritized planning* setting [33]. While it is a multi-UAV system, we plan for UAVs independently—plans between UAVs are not explictly coordinated

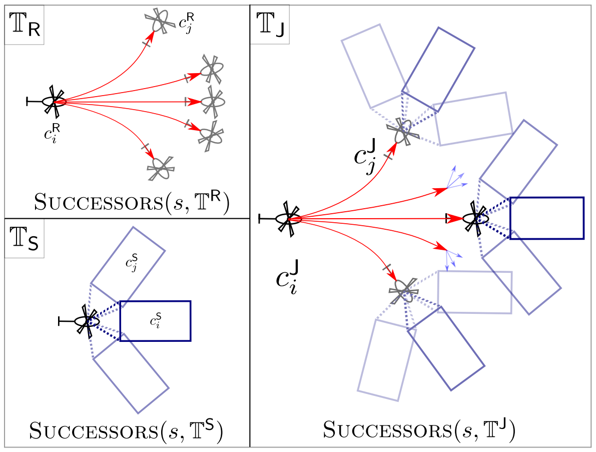


Fig. 2: An example of the successor-generation functions for the three search spaces described in Sec. III.

in [8] and is out of the scope of this paper. Our specific contribution lies in planning robot and sensor trajectories for a single UAV navigating to a goal. The framework in [8] assumes a circular sensor footprint directly underneath the UAV. In this paper, we extend the system to incorporate a rectangular footprint offset from the UAV—consequently, different sensor headings correspond to the UAV observing different areas of the environment around it.

Map. The environment map $\mathcal{M}$ consists of a priority map $\mathcal{M}^\mathsf{C}$ and a no-coverage map $\mathcal{M}^\mathrm{NC}$. The UAV must attempt to cover each cell $c_{i,j}$ at row i and column j of $\mathcal{M}^\mathsf{C}$. Such a cell is associated with two values: its lifetime $l(i,j)$ and age $a(i,j)$—the age of a cell is the time passed since the cell was last covered by a UAV, and its lifetime is a desired bound on its age. At any point of time t, $\mathcal{M}^\mathsf{C}$ holds the quantity $p(i,j) = l(i,j) - a(i,j)$ for each cell $c_{i,j}$. $\mathcal{M}^\mathsf{C}$ *decays* with time, meaning $p(i,j)$ for each cell $c_{i,j}$ reduces by one every second, thus making $c_{i,j}$ more *urgent*. Cells part of $\mathcal{M}^\mathrm{NC}$ do not need to be covered.

Sensor. In our setting, the sensor is a pan-only camera with one controllable DoF (yaw), which controls a downward-looking rectangular footprint of fixed and limited field-of-view. The area of the footprint is discretized into cells on the map according to an underlying resolution. We assume no noise in the footprint observed by the sensor.

Robot. The UAV is a kinodynamically constrained system, accounting for the robot's x and y coordinates, heading angle θ, velocity v, and timestamp t. The UAV is said to be at a cell $c_{i,j}$ if the projection of its reference point onto the xy-plane lies in cell $c_{i,j}$. A cell is said to be covered by the UAV if any point on the cell is contained in the rectangular projection of the sensor footprint on the xy-plane.

B. Problem formulation and definitions

We represent this planning problem as a search over a finite, discrete search space. Here, we define the configuration spaces of the robot (UAV) and sensor, and three state spaces that are relevant to our approaches. Each state space is associated with a set of transitions, and they together define three separate search spaces.

1) Robot state space: A feasible robot configuration is represented by $c^\mathsf{R} = (x, y, \theta, v, t)$ where x and y are the robot's $2D$ coordinates, θ is the UAV's heading, and t is the global timestamp at which this configuration is achieved (the timestamp t is part of c^R as we plan spatiotemporally collision-free trajectories for multiple robots in this framework). These five degrees of freedom together define the $5D$ robot state space $\mathbb{E}_\mathsf{R}$. A set of kinodynamically feasible motion primitives computed offline define a state lattice [34] via a set of transitions

$$\mathbb{T}_\mathsf{R} = \{(c_i^\mathsf{R}, c_j^\mathsf{R}) \mid c_i^\mathsf{R}, c_j^\mathsf{R} \in \mathbb{E}_\mathsf{R}\}$$

This defines a search space represented by a graph $\mathbb{G}_\mathsf{R}$ with nodes $\mathbb{E}_\mathsf{R}$ and edges $\mathbb{T}_\mathsf{R}$. A robot trajectory π_R is a sequence of feasible robot configurations.

2) Sensor state space: A sensor configuration is defined with respect to a corresponding robot configuration c^R as a tuple $c^\mathsf{S} = (t, \psi, H^\psi)$, where t is the timestamp in c^R, ψ is the sensor's heading angle in the global frame, and H^ψ is a list denoting the history of sensor angles assigned at all timestamps earlier than t. These state variables collectively define the sensor state space $\mathbb{E}_\mathsf{S}$, with dimensionality $(1 + |H^\psi|)$[‡]. The set of feasible sensor motions define a set of transitions

$$\mathbb{T}_\mathsf{S} = \{(c_i^\mathsf{S}, c_j^\mathsf{S}) \mid c_i^\mathsf{S}, c_j^\mathsf{S} \in \mathbb{E}_\mathsf{S}\}$$

This defines a search space represented by a graph $\mathbb{G}_\mathsf{S}$ with nodes $\mathbb{E}_\mathsf{S}$ and edges $\mathbb{T}_\mathsf{S}$. A sensor trajectory π_S is a sequence of sensor configurations.

3) Joint state space: A feasible *joint-state* configuration c^J is a concatenation of a feasible robot configuration and sensor configuration $\langle c^\mathsf{R}, c^\mathsf{S}\rangle$. These state variables collectively define the joint state space $\mathbb{E}_\mathsf{J}$ of dimensionality $(6 + |H^\psi|)$. The set of feasible transitions in $\mathbb{E}_\mathsf{J}$ is a combination of feasible transitions in $\mathbb{E}_\mathsf{R}$ and $\mathbb{E}_\mathsf{S}$

$$\mathbb{T}_\mathsf{J} = \{(c_i^\mathsf{J}, c_j^\mathsf{J}) \mid c_i^\mathsf{J}, c_j^\mathsf{J} \in \mathbb{E}_\mathsf{J}\}$$

This defines a search space represented by a graph $\mathbb{G}_\mathsf{J}$ with nodes $\mathbb{E}_\mathsf{J}$ and edges $\mathbb{T}_\mathsf{J}$. Note that since the state-lattice discretization in $\mathbb{E}_\mathsf{R}$ can be different from that in $\mathbb{E}_\mathsf{S}$, the transition set $\mathbb{T}_\mathsf{J}$ consists of actions that change robot and sensor states at their respective state discretizations.

Given these search spaces, we define the routine SUCCESSORS$(s, \mathbb{T}_\mathsf{R})$ to be the successor-generation routine for a state s that returns successor states in $\mathbb{E}_\mathsf{R}$. Similarly, we have the routines SUCCESSORS$(s, \mathbb{T}_\mathsf{S})$ and SUCCESSORS$(s, \mathbb{T}_\mathsf{J})$. Fig. 2 illustrates these three types of successors, although with much smaller branching factors for $\mathbb{T}_\mathsf{R}$ and $\mathbb{T}_\mathsf{J}$. Specifically, in $\mathbb{T}_\mathsf{J}$ for example, we generate 3 sensor-space successors for points at every 1s of a 4-second long motion primitive. We have 12 motion primitives per robot state on average, making the branching factor in the joint space $12 \times 4 \times 3 = 144$ on average. We also denote running algorithm X searching for a path from s_{start} to s_{goal} in search space $\mathbb{G}$ by X$(s_{start}, s_{goal} \mid \mathbb{G})$. For example,

[‡]Note that t is a known variable and no search is performed over it, and thus it does not contribute to the dimensionality of c^S.

A*$(s_{start}, s_{goal} \mid \mathbb{G}_\mathsf{J})$ denotes running A* search in the search space determined by $\mathbb{G}_\mathsf{J}$ (meaning state transitions are determined by $\mathbb{T}_\mathsf{J}$).

C. Cost Function

We now define the cost function associated with a transition from state s to s'. We use two costs—one associated with sensor coverage at s' where $s' \in \mathbb{E}_\mathsf{S}$ or $\mathbb{E}_\mathsf{J}$, and the other associated with the UAV's motion primitive from s to s' where $s, s' \in \mathbb{E}_\mathsf{R}$ or $\mathbb{E}_\mathsf{J}$.

Motion primitive cost. Each motion primitive is a sequence of states forward-simulated from the corresponding robot state at s, $s_{robot} = (x, y, \theta, v, t)$, following double-integrator dynamics. The cost of the primitive is equal to the time taken for the UAV to execute it. More details about the motion primitives can be found in [8].

Sensor coverage cost. For the corresponding sensor state at s', $s'_{sensor} = (t, \psi, H^\psi)$, the variables x, y, θ, ψ together define a 2D specific footprint of cells $\mathcal{F}$. Let a given footprint $\mathcal{F}$ cover $|\mathcal{F}|$ discrete cells in the map $\mathcal{M}$. Let the number of these cells lying in a coverage zone be given by $N_\mathcal{C}$ and those lying in a no-coverage zone be $N_\mathcal{NC}$:

$$N_\mathcal{C} = \sum_{i \in \mathcal{F}} \mathbb{1}\left[i \in \mathcal{M}^\mathrm{C}\right] \text{ and } N_\mathcal{NC} = \sum_{i \in \mathcal{F}} \mathbb{1}\left[i \in \mathcal{M}^\mathrm{NC}\right]$$

1) No sensor history: If we ignore the sensor history, the cost of a footprint is given by the sum of priorities of all coverage cells in $\mathcal{F}$, and an additive penalty λ scaled by the fraction of no-coverage cells in $\mathcal{F}$:

$$\mathsf{cost}_0(\mathcal{F}) = \overbrace{\sum_{i \in \mathcal{F} \wedge \mathcal{C}} p_i}^{\text{criticality measure}} + \overbrace{\lambda \times \frac{N_\mathcal{NC}}{|\mathcal{F}|}}^{\substack{\text{penalize} \\ \text{no-coverage cells}}} \tag{1}$$

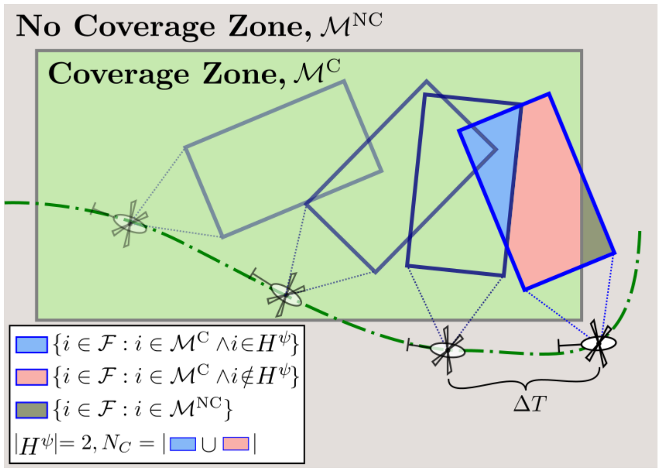
2) With sensor history: We define the following sensor coverage cost for this footprint $\mathcal{F}$ (where '$\mathbb{1}$' represents the indicator function):

$$\mathsf{cost}_H(\mathcal{F}) = \overbrace{\sum_{i \in \mathcal{F} \wedge \mathcal{C}} \underbrace{\mathbb{1}[i \notin \mathcal{H}^\psi] \times p_i}_{\text{not in history}} + \underbrace{\mathbb{1}[i \in \mathcal{H}^\psi] \times l_i}_{\text{in history}}}^{\text{criticality measure}} + \overbrace{\lambda \times \frac{N_\mathcal{NC}}{|\mathcal{F}|}}^{\substack{\text{penalize} \\ \text{no-coverage cells}}} \tag{2}$$

This cost function is illustrated in Fig. 3. Note that Eq. 2 reduces to Eq. 1 when no history is considered.

IV. Sensor Planning with Sensor History (SPLaSH)

In this section, we describe out first approach, SPLaSH. SPLaSH first quickly computes a suboptimal robot trajectory using Multi-Heuristic A* (MHA*) search in $\mathbb{G}_\mathsf{R}$. This search is performed with the motion primitive cost function (Sec. III-C). Then, it computes a sensor trajectory using (uninformed-)A* search in $\mathbb{G}_\mathsf{S}$ for a given history H. This search is performed with the sensor coverage cost function $\mathsf{cost}_H(\mathcal{F})$ (Sec. III-C). Using A* here guarantees that the sensor trajectory will be optimal for the computed robot trajectory up to the discretization and history used.



No Coverage Zone, $\mathcal{M}^\mathrm{NC}$
Coverage Zone, $\mathcal{M}^\mathrm{C}$
🟦 $\{i \in \mathcal{F} : i \in \mathcal{M}^\mathrm{C} \wedge i \in H^\psi\}$
🟥 $\{i \in \mathcal{F} : i \in \mathcal{M}^\mathrm{C} \wedge i \notin H^\psi\}$
🟩 $\{i \in \mathcal{F} : i \in \mathcal{M}^\mathrm{NC}\}$
$|H^\psi| = 2, N_C = |$🟦$\cup$🟥$|$
ΔT

Fig. 3: Pictorial explanation of our cost function $\mathsf{cost}_H(\mathcal{F})$ from Eq. 2. We consider history size, $H = 2$ in this example. For the last UAV state on the green trajectory π_R, the sensor footprint is shaded in three colours. The blue area is the overlap with previous footprints in π_R, while the red and dark green areas do not overlap. For Eq. 2, the blue area is *in history*, the red area is *not in history*, and the dark green area is *penalize no-coverage cells*. Note that for footprints too far in the past, even if there was an overlap, it has no effect.

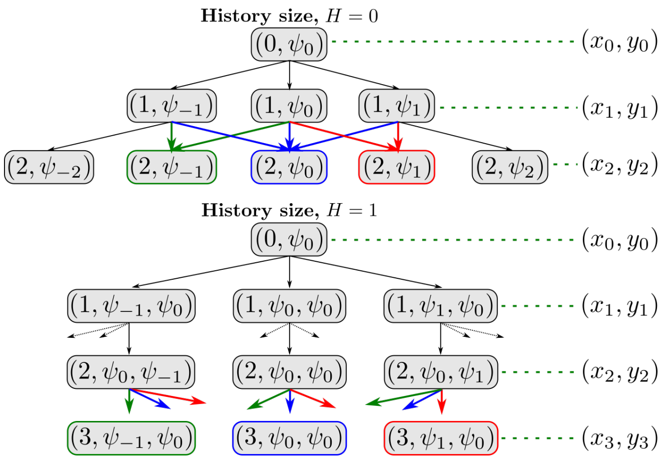


Fig. 4: Graph representation for sensor planning for history size $H = 0$ (*above*) and $H = 1$ (*below*). Each level l in the graph corresponds to a state in the UAV trajectory π_R. The search space size increases with increasing history sizes. Thus, duplicates (highlighted by coloured arrows and nodes) appear less frequently with increasing history sizes making the search for an optimal π_S more expensive.

MHA* is a variant of A* that can use multiple arbitrarily inadmissible heuristics. We omit details for brevity and refer the reader to the paper for details [35]. We use (1) a Euclidean distance heuristic, (2) a Dubin's path length heuristic, and (3) a Dijkstra's shortest path length heuristic.

Note that here and henceforth in this paper, when we mention A*, we are talking about *uninformed* A* (without a heuristic). We set aside formulating with a consistent heuristic for sensor coverage in our setting for future work. However, both SPLaSH and SPLIT both work unchanged with the addition of a heuristic. The pseudocode for SPLaSH can be found in Alg. 1.

The most important aspect of SPLaSH is accounting for sensor history in Line 3. Fig. 4 illustrates the effect of history values $H = 0$ and $H = 1$ on the search graph $\mathbb{G}_\mathsf{S}$ for a given

Algorithm 1 Sensor Planning with Sensor History (SPLASH)

```
Input: $s_{start}$, $s_{goal}$, $\mathsf{H}$
Output: $\pi_f$ (final trajectory in joint space)
1: procedure MAIN()
2:     $\pi_{robot} \leftarrow$ MHA*($s_{start}$, $s_{goal}$ | $\mathbb{G}_\mathsf{R}$)
                              ▷ using motion primitive cost as in Sec. III-C
3:     $\pi_{sensor} \leftarrow$ A*($s_{start}$, $s_{goal}$ | $\mathbb{G}_\mathsf{S}$) with $\mathsf{H}$ states in sensor history
                  ▷ using sensor coverage cost, $\mathsf{cost}_H(\mathcal{F})$ as in Sec. III-C
4:     $\pi_{joint} \leftarrow$ concatenate $\pi_{robot}$ and $\pi_{sensor}$
                              ▷ creates a joint-space plan as in Sec. III-B.3
5:     return $\pi_{joint}$
```

initial sensor heading ψ_0. Each level in Fig. 4 corresponds to a waypoint along the robot trajectory π_R. The figure denotes state $c^\mathsf{S} = (t, \psi, H^\psi)$ as a tuple where H^ψ is the last H elements in the tuple. For any state c^S, the sensor angle can either be changed by one step (increment or decrement), or it may remain the same.

The effect of H values is illustrated by the colored arrows and vertices in the graph—two arrows of the same color end up at a unique state in the graph. The key idea is as follows: For $H = 0$, ending up at ψ_0 on level 2 is considered the same state, whether you come from ψ_0 or ψ_{-1} or ψ_1—we only care about the *current* sensor angle. But for $H = 1$, ending up at ψ_0 on level 2 is considered a different state in all these three cases because we maintain 1 historical sensor angle. This can be incorporated by defining the state as $c^\mathsf{S} = (t, \psi, H^\psi)$ where $|H^\psi| = 1$. Observe that states are replicated in this way a lot more frequently for $H = 0$ than for $H = 1$, meaning the graph for $H = 0$ has much (in fact, exponentially) lesser states than that for $H = 1$.

V. Sensor Planning with Local Iterative Tunneling (SPLIT)

SPLASH takes into account sensor history and incentivizes the search to compute plans where overlaps are minimized. However, it operates with a fixed, suboptimal robot trajectory that optimizes only motion primitive cost. Recall that it is empirically observed that approximate search algorithms tend to overlook better solutions that are actually "close" to the computed solution in the space of solution paths [7]. The final solution that SPLASH gives us—say π—is most likely suboptimal with respect to the *coverage* cost in the space of joint-space solutions. **S**ensor **P**lanning with **L**ocal **I**terative **T**unneling (SPLIT) locally refines π by performing searches in small search spaces around π in the joint space using the sensor coverage cost function $\mathsf{cost}_H(\mathcal{F})$ (Sec. III-C), increasing in size with each iteration. We call these search spaces *tunnels*, and this is an application of the ITSA* algorithm [7].

We provide pseudocode for SPLIT in Alg. 2. Lines in blue indicate the differences from standard A* search. Line 2 obtains the initial solution from SPLASH. Then, LOCALITERATIVETUNNELING refines this solution locally by performing A* searches in iterative tunnels. The "level" of a state s corresponds to the distance from the initial path π_i to s computed as the smallest number of edges on a path

Algorithm 2 Sensor Planning with Local Iterative Tunneling (SPLIT)

```
Input: $s_{start}$, $s_{goal}$, $\mathsf{T_{overall}}$ (time limit)
Output: $\pi_f$ (final trajectory in joint space)
1: procedure MAIN()
2:     $\pi_i \leftarrow$ SPLASH($s_{start}, s_{goal}, \mathsf{H} = 0$)          ▷ Initialization step
3:     $\mathsf{t_{SPLASH}} \leftarrow$ time taken for SPLASH to terminate
4:     $\pi_f \leftarrow$ LOCALITERATIVETUNNELING($\pi$, $\mathsf{T_{overall}} - \mathsf{t_{SPLASH}}$)
                                                        ▷ Refinement step
5:     return $\pi_f$

6: procedure LOCALITERATIVETUNNELING($\pi_i$, t)
7:     iteration ← 1
8:     Create and store all states $s_i \in \pi_i$ in memory with $level(s_i) = 0$
9:     while time t remains do                          ▷ Iterative Tunneling loop
10:        $s_{start} \leftarrow$ first state in $\pi$
11:        $s_{goal} \leftarrow$ last state in $\pi$
12:        $g(s_{goal}) = \infty$; $g(s_{start}) = 0$
13:        $bp(s_{start}) = bp(s_{goal})$ = NULL
14:        Insert $s_{start}$ into OPEN with KEY($s_{start}$)
15:        while OPEN not empty do                          ▷ Modified A* loop
16:            $s \leftarrow$ OPEN.MIN()                   ▷ where OPEN is a min-heap
17:            if $s$ is goal then
18:                Backtrack from $s$ to obtain solution $\pi_f$
19:                break
20:            for successor $s'$ in SUCCESSORS($s$, $\mathbb{T}_\mathsf{J}$) do
                           ▷ successors are computed via transitions in $\mathbb{E}_\mathsf{J}$
21:                if $s'$ not closed then
22:                    if $g(s') > g(s) + c(s, s')$ then
23:                        $g(s') = g(s) + c(s, s')$
                                ▷ where $c(s, s')$ is the sensor coverage cost
24:                        $bp(s') = s$
25:                        $level(s') = level(s) + 1$
26:                    if $level(s') \leq$ iteration then
27:                        Insert $s'$ into OPEN with KEY($s'$)
28:        iteration ← iteration + 1
29:        t ← t − time elapsed in current iteration
30:    return $\pi_f$

31: procedure KEY($s$)
32:    return $g(s) + h(s)$            ▷ where $h(.)$ is a consistent heuristic
```

from any state on π_i to s [7]. In the beginning, every state on the initial plan π_i is stored in memory with level 0.

The refinement process is essentially A* being performed repeatedly, with the addition of lines 25–27. The level of any newly generated state is set to one more than the level of its parent. Only a state whose level is lesser than the current iteration number is inserted into the OPEN list. This is what creates tunnels increasing in size per iteration. LOCALITERATIVETUNNELING—and consequently, SPLIT—terminates when the time available for local refinement runs out.

VI. Experimental Results

We evaluate our approaches by running them over 10 randomly generated start-goal pairs per map for 20 maps. We pick maps as seen in the persistent coverage framework described in [8] (Sec. III). The maps are generated by letting the map in the framework decay for several minutes while one UAV with a fixed sensor covers it persistently and pick snapshots of the map at different points in time, giving us maps with complex coverage zones.

Evaluation. Let a given trajectory in the joint space cover N cells that lie in coverage zones. Let the quantity $\sum_i p_i$ denote the sum of priorities of all such cells. We value

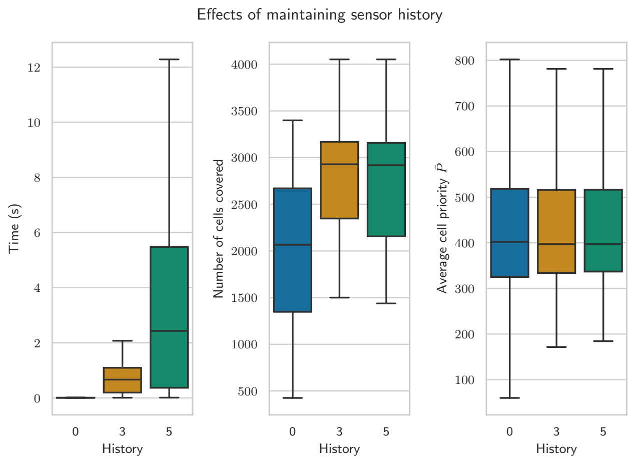


Fig. 5: Results of running SPLASH for sensor histories of size $0, 3, 5$.

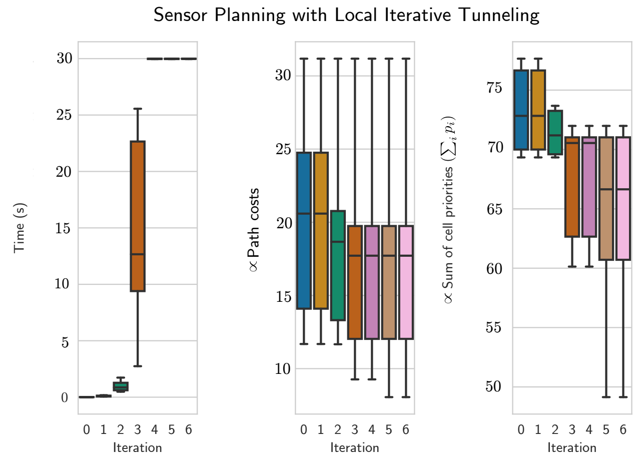


Fig. 6: Results of running SPLIT timed out at 30s.

two things: covering a large number of cells, and covering important cells (those with a low priority value). Thus, a large value of N and low values of $\sum_i p_i$ are desirable for a given plan. The quantity $\bar{P} = \frac{\sum_i p_i}{N}$ denotes the average of the priority values of all such cells. A low value of $\bar{P}$ is not necessarily an indicator of a desirable plan due to various cell priorities encountered in a complex map. For any two plans having the same value of $\bar{P}$, it depends on whether the user prefers covering more cells or important cells. As an illustration, consider plan A that covers only one cell with priority $\{10\}$, and plan B that covers 6 cells with priorities $\{5, 5, 5, 5, 20, 20\}$. The average priority of cells in both plans is the same $(= 10)$, and it is up to the user to decide which plan is considered "better".

Since SPLASH penalizes footprint overlaps, we see an increase in the number of cells covered as a larger sensor history is maintained (see Fig. 5). We also see that maintaining a sensor history of size 5 gives us no more value than size 3 in practical settings. Also notice that $\bar{P}$ stays fairly unchanged over several trajectories. This can be attributed to sufficiently complex maps have many different priority values for cells and no single, contiguous coverage zone—maintaining sensor history would indeed lead to covering more cells, but the average priority over these cells would be approximately the same.

Since SPLIT refines the trajectory locally to optimize coverage cost, we see an increase in information gain, or a decrease in $\sum_i p_i$, with each iteration (see Fig. 6). We also see decreasing path costs (g-value of the goal) with each iteration. Note that its performance largely depends on the immediate area around the initial plan which will be explored in the iterative tunnels. If this immediate area has only a few more important cells to cover, the refined plan will largely stay the same.

A natural baseline is to search directly in the joint space of robot and sensor variables. This requires a cost function that is a linear combination of the motion primitive and sensor coverage cost. Running MHA* on these start-goal pairs with such a cost function yielded an average planning time of 8.44 ± 6.40s—significantly larger than running SPLASH with a sensor history of size 3 or SPLIT for 2 or 3 iterations. (We set a timeout of 20s while obtaining this value, so this is a conservative estimate and true value is in fact larger.)

Note that iteration 4 onward, SPLIT takes a long time to terminate. This is useful for real-world planning problems only if the planner is given sufficient time. For example, in the framework that we have built upon [8], it is possible to tune the duration of a committed UAV trajectory. For a longer duration, we have more time available to plan to the next goal assigned to the UAV. Moreover, these results show that the solution can still improve after the third iteration and that a local minimum is not encountered by then.

VII. Conclusion and Future Work

We present two search-based approaches for generating robot and sensor trajectories in goal-directed 2D coverage tasks, namely **S**ensor **Pla**nning with **S**ensor **H**istory (SPLASH) and **S**ensor **Pl**anning with **L**ocal **I**terative **T**unneling (SPLIT). SPLASH solves for robot and sensor trajectories independently in state spaces while maintaining a history of sensor headings. SPLIT is a two-step approach that refines this solution by searching its local neighborhood in the joint space for a better solution. We show that SPLASH is a practical alternative to to running standard search-based planning in the full joint space of robot and sensor state variables, and that SPLIT can be used to further refine the solution computed by SPLASH given enough time.

A limitation of ITSA*, and consequently of SPLIT, is that the A* searches do not reuse any search efforts between subsequent iterations, and so each iteration takes longer than the last. Reusing search efforts between iterations will lead to considerable speed-ups, leading to faster refinement of the trajectory. Further, we do not look into maintaining sensor histories within SPLIT. It can be useful to adaptively increase the size of the sensor history maintained with increasing iterations in SPLIT. We set aside these two limitations as opportunities for future work.

References

[1] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[2] A. Nedjati, G. Izbirak, B. Vizvari, and J. Arkat, "Complete coverage path planning for a multi-UAV response system in post-earthquake assessment," *Robotics*, vol. 5, no. 4, p. 26, 2016.

[3] R. N. Smith, M. Schwager, S. L. Smith, B. H. Jones, D. Rus, and G. S. Sukhatme, "Persistent ocean monitoring with underwater gliders: Adapting sampling resolution," *JFR*, vol. 28, no. 5, pp. 714–741, 2011.

[4] S. Srinivasan, H. Latchman, J. Shea, T. Wong, and J. McNair, "Airborne traffic surveillance systems: video surveillance of highway traffic," in *Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks*, 2004, pp. 131–135.

[5] L. Teixeira, I. Alzugaray, and M. Chli, "Autonomous aerial inspection using visual-inertial robust localization and mapping," in *FSR*. Springer, 2018, pp. 191–204.

[6] I. Pohl, "Heuristic search viewed as path finding in a graph," *Artificial intelligence*, vol. 1, no. 3-4, pp. 193–204, 1970.

[7] D. Furcy, "ITSA*: Iterative tunneling search with A*," in *Proceedings of the National Conference on Artificial Intelligence (AAAI), Workshop on Heuristic Search, Memory-Based Heuristics and Their Applications, Boston, Massachusetts*, 2006.

[8] T. Kusnur, S. Mukherjee, D. M. Saxena, T. Fukami, T. Koyama, O. Salzman, and M. Likhachev, "A planning framework for persistent, multi-uav coverage with global deconfliction," in *2019 International Conference on Field and Service Robotics (FSR)*, 2019.

[9] A. O. Hero and D. Cochran, "Sensor management: Past, present, and future," *IEEE Sensors Journal*, vol. 11, no. 12, pp. 3064–3075, 2011.

[10] V. Gupta, T. H. Chung, B. Hassibi, and R. M. Murray, "On a stochastic sensor selection algorithm with applications in sensor scheduling and sensor coverage," *Automatica*, vol. 42, no. 2, pp. 251–260, 2006.

[11] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an rgb-d camera," in *Robotics Research*. Springer, 2017, pp. 235–252.

[12] P. Furgale and T. D. Barfoot, "Visual teach and repeat for long-range rover autonomy," *Journal of Field Robotics*, vol. 27, no. 5, pp. 534–560, 2010.

[13] S. Scherer, L. Chamberlain, and S. Singh, "Autonomous landing at unprepared sites by a full-scale helicopter," *Robotics and Autonomous Systems*, vol. 60, no. 12, pp. 1545–1562, 2012.

[14] Q. Du, V. Faber, and M. Gunzburger, "Centroidal voronoi tessellations: Applications and algorithms," *SIAM review*, vol. 41, no. 4, pp. 637–676, 1999.

[15] H. Mori, "Active sensing in vision-based stereotyped motion," in *EEE International Workshop on Intelligent Robots and Systems, Towards a New Frontier of Applications*. IEEE, 1990, pp. 167–174.

[16] G. Costante, C. Forster, J. Delmerico, P. Valigi, and D. Scaramuzza, "Perception-aware path planning," *arXiv preprint arXiv:1605.04151*, 2016.

[17] L. Meier, J. Peschon, and R. Dressler, "Optimal control of measurement subsystems," *IEEE Transactions on Automatic Control*, vol. 12, no. 5, pp. 528–536, 1967.

[18] Y. Oshman, "Optimal sensor selection strategy for discrete-time state estimators," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 30, no. 2, pp. 307–314, 1994.

[19] T. Mukai and M. Ishikawa, "An active sensing method using estimated errors for multisensor fusion systems," *IEEE Transactions on Industrial Electronics*, vol. 43, no. 3, pp. 380–386, 1996.

[20] T. H. Chung, V. Gupta, J. W. Burdick, and R. M. Murray, "On a decentralized active sensing strategy using mobile sensor platforms in a network," in *2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601)*, vol. 2. IEEE, 2004, pp. 1914–1919.

[21] S. Kagami and M. Ishikawa, "A sensor selection method considering communication delays," *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, vol. 89, no. 5, pp. 21–31, 2006.

[22] F. Bourgault, T. Furukawa, and H. F. Durrant-Whyte, "Coordinated decentralized search for a lost target in a bayesian world," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 1. IEEE, 2003, pp. 48–53.

[23] A. Ryan and J. K. Hedrick, "Particle filter based information-theoretic active sensing," *Robotics and Autonomous Systems*, vol. 58, no. 5, pp. 574–584, 2010.

[24] S. Arora and S. Scherer, "Pasp: Policy based approach for sensor planning," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3479–3486.

[25] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas, "Information acquisition with sensing robots: Algorithms and error bounds," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6447–6454.

[26] B. Schlotfeldt, N. Atanasov, and G. J. Pappas, "Maximum information bounds for planning active sensing trajectories," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4913–4920.

[27] Y. Kantaros, B. Schlotfeldt, N. Atanasov, and G. J. Pappas, "Asymptotically optimal planning for non-myopic multi-robot information gathering." in *Robotics: Science and Systems*, 2019.

[28] W. Lu, G. Zhang, and S. Ferrari, "An information potential approach to integrated sensor path planning and control," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 919–934, 2014.

[29] P. Váňa and J. Faigl, "On the dubins traveling salesman problem with neighborhoods," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 4029–4034.

[30] J. Faigl, P. Váňa, M. Saska, T. Báča, and V. Spurný, "On solution of the dubins touring problem," in *2017 European Conference on Mobile Robots (ECMR)*. IEEE, 2017, pp. 1–6.

[31] R. Pěnička, J. Faigl, and M. Saska, "Physical orienteering problem for unmanned aerial vehicle data collection planning in environments with obstacles," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 3005–3012, 2019.

[32] R. Pěnička, J. Faigl, M. Saska, and P. Váňa, "Data collection planning with non-zero sensing distance for a budget and curvature constrained unmanned aerial vehicle," *Autonomous Robots*, vol. 43, no. 8, pp. 1937–1956, 2019.

[33] M. Erdmann and T. Lozano-Perez, "On multiple moving objects," *Algorithmica*, vol. 2, no. 1-4, p. 477, 1987.

[34] M. Pivtoraiko and A. Kelly, "Generating near minimal spanning control sets for constrained motion planning in discrete state spaces," in *IROS*, 2005, pp. 3231–3237.

[35] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic A*," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 224–243, 2016.