Perception-Based Temporal Logic Planning in Uncertain Semantic Maps

Yiannis Kantaros, *Member, IEEE*, Samarth Kalluraya, Qi Jin, and George J. Pappas, *Fellow, IEEE*

***Abstract*—In this article, we address a multi-robot planning problem in environments with partially unknown semantics. The environment is assumed to have a known geometric structure (e.g., walls) and to be occupied by static labeled landmarks with uncertain positions and classes. This modeling approach gives rise to an uncertain semantic map generated by semantic simultaneous localization and mapping algorithms. Our goal is to design control policies for robots equipped with noisy perception systems so that they can accomplish collaborative tasks captured by global temporal logic specifications. To specify missions that account for environmental and perceptual uncertainty, we employ a fragment of linear temporal logic (LTL), called co-safe LTL, defined over perception-based atomic predicates modeling probabilistic satisfaction requirements. The perception-based LTL planning problem gives rise to an optimal control problem, solved by a novel sampling-based algorithm, that generates open-loop control policies that are updated online to adapt to a continuously learned semantic map. We provide extensive experiments to demonstrate the efficiency of the proposed planning architecture.**

***Index Terms*—Motion planning, multi-robot systems, sensor-based and reactive planning.**

I. Introduction

AUTONOMOUS robot navigation is a fundamental problem that has received considerable research attention [1]. The basic motion planning problem consists of generating robot trajectories that reach a known desired goal region starting from an initial configuration while avoiding known obstacles. More recently, a new class of planning approaches has been developed that can handle a richer class of tasks than the classical point-to-point navigation and can capture temporal and Boolean specifications. Such tasks can be, e.g., sequencing or coverage [2], data gathering [3], or persistent surveillance [4] and can be modeled using formal languages, such as linear temporal logic (LTL) [5], [6]. Several control synthesis methods have been proposed to design correct-by-construction controllers that satisfy temporal logic specifications assuming robots with known dynamics that operate in known environments [7]–[15]. As a result, these methods cannot be applied to scenarios where the environment is initially unknown, and therefore, online replanning may be required as environmental maps are constructed, resulting in limited applicability. To mitigate this issue, learning-based approaches have also been proposed that consider robots with unknown dynamics operating in fully or partially unknown environments. These approaches learn policies that directly map on-board sensor readings to control commands in a trial-and-error fashion; see, e.g., [16]–[19]. However, learning-based approaches, in addition to being data inefficient, are specialized to the environment and the robotic platform they were trained on.

In this article, we address a motion planning problem for a team of mobile sensing robots with known dynamics that are responsible for accomplishing collaborative tasks, expressed using a fragment of LTL, called co-safe LTL, in the presence of environmental and sensing uncertainty. Specifically, the environment is assumed to have known geometry, while being occupied by static landmarks with uncertain labels/classes (*semantic* uncertainty) located at uncertain positions (*metric* uncertainty) giving rise to an uncertain semantic map with known geometry; see, e.g., Fig. 1. The semantic map is determined by Gaussian and arbitrary discrete distributions over the positions and labels of the landmarks, respectively. The robots are equipped with noisy/imperfect perception systems (e.g., cameras and object detectors and classifiers) allowing them to take positional and class measurements about the landmarks. To define mission requirements that account for perceptual and environmental uncertainty, similar to [20] and [21], we specify temporal logic tasks defined over perception-based probabilistic predicates. Specifically, we incorporate probabilistic satisfaction requirements at the atomic-predicate level as opposed to probabilistic or metric/signal temporal logic languages that evaluate the probabilistic or the robust satisfaction of temporal logic formulas [22]–[26]. This way, we allow a user to put different weights on the individual components/requirements comprising an LTL formula (e.g., staying away from a hostile patrolling agent may be more important than reaching a goal destination, which cannot be captured by probabilistic temporal

Manuscript received 21 June 2021; revised 23 October 2021; accepted 28 December 2021. Date of publication 9 February 2022; date of current version 8 August 2022. This work was supported in part by the Air Force Research Laboratory, in part by the Defense Advanced Research Projects Agency under Grant FA8750-18-C-0090, and in part by the Army Research Laboratory's Distributed and Collaborative Intelligent Systems and Technology Collaborative Research Alliance under Grant W911NF-17-2-0181. This paper was recommended for publication by Associate Editor P. M. Wensing and Editor W. Burgard upon evaluation of the reviewers' comments. *(Corresponding author: Yiannis Kantaros.)*

Kantaros and Samarth Kalluraya were with the Department of Computer and Information Science and the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104 USA. They are now with the Department of Electrical and Systems Engineering, Washington University in St. Louis, St. Louis, MO 63112 USA (e-mail: ioannisk@wustl.edu; samarth3@seas.upenn.edu).

Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: qijin1@seas.upenn.edu; pappasg@seas.upenn.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TRO.2022.3144073.

Digital Object Identifier 10.1109/TRO.2022.3144073

1552-3098 © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:45:49 UTC from IEEE Xplore. Restrictions apply.

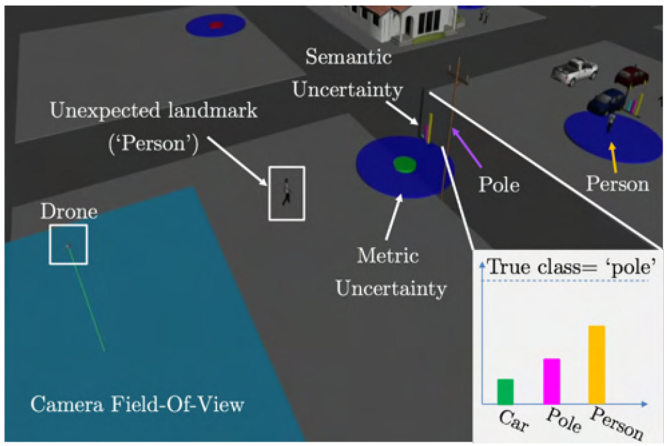


Fig. 1. A drone equipped with a noisy downward facing camera (blue rectangle), and an object detector is responsible for delivering supplies to uncertain ground units (e.g., people and cars) in a specific order while always avoiding colliding with uncertain poles and flying over known residential areas. The environment is modeled as an uncertain semantic map with known geometry consisting of static labeled landmarks (e.g., poles, people, or parked cars) with uncertain positions (metric uncertainty) and labels (semantic uncertainty). Each landmark is associated with a prior Gaussian and a discrete distribution modeling its positional and class uncertainty, respectively, while unknown landmarks without prior information may exist as well.

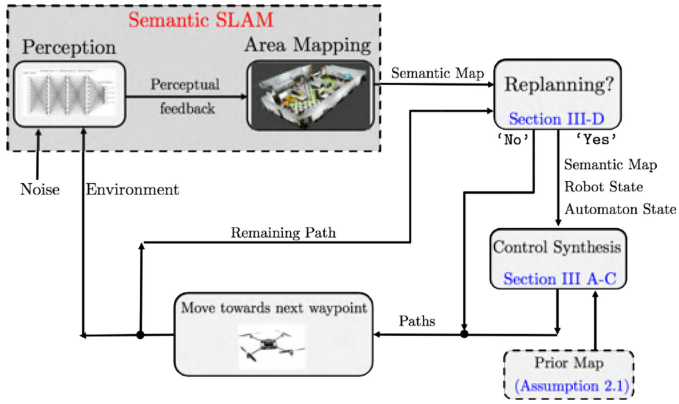


Fig. 2. Graphical depiction of the proposed architecture for planning over continuously learned uncertain semantic maps. First, given a mission specification ϕ and an uncertain prior belief about the environment, initial paths are constructed. The robots follow the designed paths while updating the semantic map by fusing the collected measurements. Replanning is triggered occasionally to adapt planning to the continuously learned map.

logic languages straightforwardly). Then, our goal is to design perception-based control policies so that the multi-robot system satisfies the co-safe LTL specification.

The problem of synthesizing controllers in the presence of mapping or localization uncertainty typically gives rise to partially observable Markov decision processes (POMDPs) [21], [27] that suffer from the curse of dimensionality and history. To avoid the need for computationally expensive POMDP methods, we formulate the planning problem as a deterministic optimal control problem that designs open-loop perception-based control policies that satisfy the assigned specification. To solve this problem, building upon existing sampling-based planners [28]–[30], we present a sampling-based approach that, given an uncertain prior belief about the semantic map, explores the robot motion space, a (sub)space of map uncertainty, and an automaton space corresponding to the assigned mission. During the offline control synthesis process, the robots explore the metric environmental uncertainty, i.e., predict how the metric uncertainty will change based on their future control actions under linearity and Gaussian assumptions on the sensor models [31], [32]. Prediction of the evolution of semantic uncertainty in the offline control synthesis phase is not possible, as it requires label measurements (i.e., images), which are impossible to obtain *a priori*/offline. To ensure that the proposed sampling-based approach can efficiently explore this large hybrid space, we adopt the biased sampling strategy, developed in our previous work [30], that biases exploration toward regions that are expected to be informative and contribute to satisfaction of the assigned specification. We show that the proposed sampling-based algorithm is probabilistically complete and asymptotically optimal under Gaussian and linearity assumptions on the sensor models.

As the robots execute the designed control policies, they take measurements that allow them to update their belief about both the metric and the semantic uncertainty of the environment using recently proposed semantic simultaneous localization and mapping (SLAM) algorithms [33], [34]. To adapt to the online learned map, the robots design new paths by re-applying the sampling-based approach. Replanning occurs only at time instants, where the offline predicted map differs significantly from the online constructed map. Formally, the replanning time instants are determined online by an automaton corresponding to the assigned co-safe LTL formula. The mission terminates when an accepting condition of this automaton is satisfied. The proposed planning architecture over continuously learned semantic maps is also summarized in Fig. 2. Extensions of the proposed algorithm to account for nonlinear sensor models and environments with no prior metric/semantic information are discussed as well. We provide extensive experiments that corroborate the theoretical analysis and show that the proposed algorithm can address large-scale planning tasks under environmental and sensing uncertainty.

A. Related Research

1) Sampling-Based Temporal Logic Planning: Sampling-based approaches for temporal logic planning have also been proposed in [14], [15], and [35]–[38], but they consider *known* environments. Typically, these works build upon sampling-based planning algorithms originally developed for reachability tasks [39]. Specifically, in [35], a sampling-based algorithm is proposed, which incrementally builds a Kripke structure until it is expressive enough to generate a path that satisfies a task specification expressed in deterministic μ-calculus. In fact, Karaman and Frazzoli[35] build upon the rapidly-exploring random graph (RRG) algorithm [39] to construct an expressive enough abstraction of the environment . To enhance scalability of [35], a sampling-based temporal logic path planning algorithm is proposed in [14], which also builds upon the RRG algorithm but constructs incrementally sparse graphs to build discrete abstractions of the workspace and the robot dynamics. These

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:45:49 UTC from IEEE Xplore. Restrictions apply.

abstractions are, then, composed with an automaton, capturing the task specification, which yields a product automaton. Then, correct-by-construction paths are synthesized by applying graph search methods on the product automaton. However, similar to [35], as the number of samples increases, the sparsity of the constructed graph is lost, and as a result, this method does not scale well to large planning problems either. Similar abstraction-free temporal logic planning algorithms have been presented in [36]–[38] as well. A highly scalable sampling-based approach for LTL planning problems is also presented in [15] that can handle hundreds of robots, which, unlike the previous works, requires user-specified discrete abstractions of the robot dynamics [40]. Note that unlike these works, as discussed earlier, our sampling-based approach also explores a map uncertainty space to account for environmental and sensing uncertainty.

2) Temporal Logic Planning Under Map Uncertainty: Temporal logic planning in the presence of map uncertainty in terms of incomplete environment models is considered in [41]–[47]. Common in these works is that they consider static environments with unknown geometry but with known metric/semantic structure. As a result, in these works, the regions of interest over which the LTL missions are defined are *a priori* known. Additionally, to model these partially unknown environments, transition systems [5] or occupancy grid maps [48] are used that are continuously updated assuming perfect sensor feedback; as a result, LTL tasks are defined using deterministic atomic predicates. To the contrary, in this article, we consider environments with known geometry but with uncertain metric and semantic structure, i.e., the locations and the labels of regions/objects of interest are uncertain. To account for this novel environmental uncertainty, we model the environment as an uncertain semantic map—which is continuously learned through uncertain perceptual feedback—and we introduce probabilistic atomic predicates over the uncertain map to define robot missions. For instance, Guo *et al.* [41], [42] model the environment as transition systems, which are partially known. Then, discrete controllers are designed by applying graph search methods on a product automaton. As the environment, i.e., the transition system, is updated assuming perfect sensor feedback, the product automaton is locally updated as well, and new paths are redesigned by applying graph search approaches on the revised automaton. Common in all these works is that, unlike our approach, they rely on discrete abstractions of the robot dynamics [40], [49]. Thus, correctness of the generated motion plans is guaranteed with respect to the discrete abstractions resulting in a gap between the generated discrete motion plans and their physical low-level implementation [50]. An abstraction-free reactive method for temporal logic planning over continuously learned occupancy grid maps is proposed in [47]. Note that Kantaros *et al.* [47] can handle sensing and environmental uncertainty, as shown experimentally, but completeness guarantees are provided assuming perfect sensors, which is not the case in this article.

The most relevant works are presented in [30] and [51] that also address temporal logic motion planning problems in uncertain environments modeled as uncertain semantic maps. In particular, Fu *et al.* [51] consider sequencing tasks, captured by co-safe LTL formulas, under the assumption of landmarks with uncertain locations but *a priori* known labels. This article is extended in [30] by considering landmarks with uncertain locations and labels and task specifications captured by arbitrary co-safe temporal logic formulas. Common in both works is that control policies are designed based on a given semantic map—which is never updated—without considering the sensing capabilities of the robots. In contrast, in this article, the proposed algorithm designs *perception-based* control policies that actively decrease the metric uncertainty in the environment to satisfy probabilistic satisfaction requirements that are embedded into the mission specification. In other words, the proposed method may find feasible paths even if such paths do not exist for the initial uncertain semantic map. In addition, unlike [30] and [51], we show that the proposed algorithm can be extended to environments with no prior metric/semantic information. Finally, we provide formal completeness and optimality guarantees that do not exist in [30] and [51].

B. Contribution

The contribution of this article can be summarized as follows. *First*, we formulate the first perception-based LTL planning problem over uncertain semantic maps that are continuously learned by semantic SLAM methods. In fact, we propose the first formal bridge between temporal logic planning and semantic SLAM methods. *Second*, we present a sampling-based approach that can synthesize perception-based control policies for robot teams that are tasked with accomplishing temporal logic missions in uncertain semantic environments. *Third*, we show that the proposed sampling-based approach is probabilistically complete and asymptotically optimal under linearity and Gaussian assumptions. *Fourth*, we show that the proposed algorithm can be applied to environments with no prior semantic/metric information. *Fifth*, we provide extensive experiments that illustrate the efficiency of the proposed algorithm and corroborate its theoretical guarantees.

II. Problem Definition

A. Multirobot System

Consider N mobile robots governed by the following dynamics:

$$\mathbf{p}_j(t+1) = \mathbf{f}_j(\mathbf{p}_j(t), \mathbf{u}_j(t)) \tag{1}$$

for all $j \in \mathcal{N} := \{1, \ldots, N\}$. In (1), $\mathbf{p}_j(t) \in \mathbb{R}^n$ stands for the state (e.g., position and orientation) of robot j in an environment $\Omega \subseteq \mathbb{R}^m, m \in \{2, 3\}$, at discrete time t.[1] In addition, $\mathbf{u}_j(t) \in \mathcal{U}_j$ stands for a control input applied at time t by robot j, where $\mathcal{U}_j$ denotes a *finite* space of admissible control inputs. Hereafter, we compactly model the team of robots, each equipped with a finite library of motion primitives, as

$$\mathbf{p}(t+1) = \mathbf{f}(\mathbf{p}(t), \mathbf{u}(t)) \tag{2}$$

where $\mathbf{p}(t) \in \Omega^N$, $\forall t \geq 0$, and $\mathbf{u}(t) \in \mathcal{U} := \mathcal{U}_1 \times \cdots \times \mathcal{U}_N$.

[1]For simplicity of notation, hereafter, we also denote $\mathbf{p}_j(t) \in \Omega$.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:45:49 UTC from IEEE Xplore. Restrictions apply.

B. Uncertain Semantic Maps

The robots operate in an environment with a known geometric structure (e.g., walls) but with uncertain metric and semantic structure. Specifically, the obstacle-free space $\Omega_{\text{free}} \subseteq \Omega$ is assumed to be known, while Ω_{free} is cluttered with $M > 0$ uncertain, static, and labeled landmarks ℓ_i giving rise to an uncertain semantic map $\mathcal{M} = \{\ell_1, \ell_2, \ldots, \ell_M\}$. Each landmark $\ell_i = \{\mathbf{x}_i, c_i\} \in \mathcal{M}$ is defined by its position $\mathbf{x}_i \in \Omega$ and its class $c_i \in \mathcal{C}$, where $\mathcal{C}$ is a finite set of semantic classes. The robots do not know the true landmark positions and classes, but instead they have access to a probability distribution over the space of all possible maps. Such a distribution can be produced by recently proposed semantic SLAM algorithms [33], [34] and typically consists of a Gaussian distribution over the landmark positions, modeling metric uncertainty in the environment, and an arbitrary discrete distribution over the landmark class modeling semantic uncertainty. Specifically, we assume that $\mathcal{M}$ is determined by parameters $(\hat{\mathbf{x}}_i, \Sigma_i, d_i)$, for all landmarks ℓ_i, such that $\mathbf{x}_i \sim \mathcal{N}(\hat{\mathbf{x}}_i, \Sigma_i)$, where $\hat{\mathbf{x}}_i$ and Σ_i denote the mean and covariance matrix of the position of landmark ℓ_i, respectively, and $c_i \sim d_i$, where $d_i : \mathcal{C} \to [0, 1]$ is an arbitrary discrete distribution over the finite set of possible classes. Hereafter, we compactly denote by $(\hat{\mathbf{x}}, \Sigma, d)$ the parameters that determine all landmarks in the semantic map $\mathcal{M}$.

In what follows, we assume that an uncertain prior belief about the metric/semantic structure of the environment is available by a user to the robots, which can be updated online using SLAM methods. A discussion on relaxing this assumption is provided in Section IV.

Assumption 2.1 (Uncertain prior map): We assume that initially the robots have access to Gaussian and discrete distributions denoted by $\mathbf{x}_i \sim \mathcal{N}(\hat{\mathbf{x}}_i(0), \Sigma_i(0))$ and $c_i \sim d_i(0)$, respectively, for all landmarks $\ell_i \in \mathcal{M}$, where the number of landmarks and the set of available classes are *a priori* known.

Example 2.2 (Semantic map): An example of an uncertain semantic map is illustrated in Fig. 3 that consists of $M = 7$ landmarks with set of classes $\mathcal{C} = \{$"person," "door," "supplies"$\}$. Uncertain semantic maps provide a novel means to define mission objectives in terms of meaningful semantic objects in uncertain environments. For instance, in Fig. 3, consider a task that requires the robots to collect all available supplies and deliver them in a specific order to injured people.

C. Perceptual Capabilities

The robots are equipped with sensors (e.g., cameras) to collect measurements associated with the landmark positions $\mathbf{x}$ as per the following *linear observation model*: $\mathbf{y}_j(t) = \mathbf{M}_j(\mathbf{p}_j(t))\mathbf{x} + \mathbf{v}_j(t)$, where $\mathbf{y}_j(t)$ is the measurement signal at discrete time t taken by robot $j \in \mathcal{N}$. For instance, linear observation models with respect to the hidden state have been considered in [52]. In addition, $\mathbf{v}_j(t) \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_j)$ is sensor-state-dependent Gaussian noise with covariance $\mathbf{R}_j$. Hereafter, we compactly denote the observation models of all robots as

$$\mathbf{y}(t) = \mathbf{M}(\mathbf{p}(t))\mathbf{x} + \mathbf{v}(t), \quad \mathbf{v}(t) \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \tag{3}$$

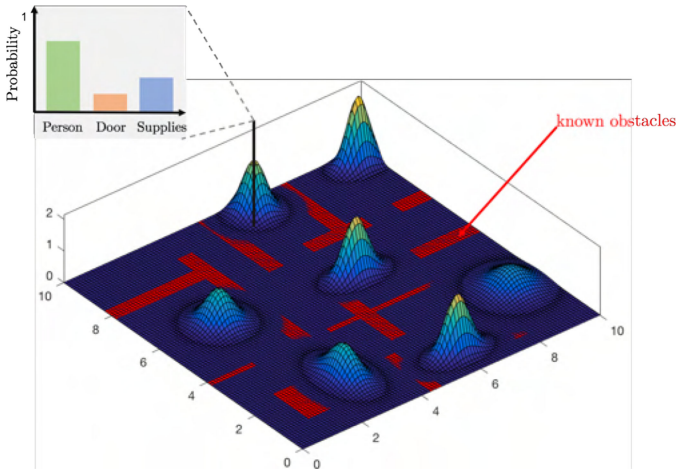


Fig. 3. Environment with a known geometric structure modeled as an uncertain semantic map consisting of $M = 7$ landmarks with a set of semantic classes defined as $\mathcal{C} = \{$"person," "door," "supplies"$\}$. Each landmark is determined by a Gaussian distribution and a discrete distribution modeling uncertainty in its position and class, respectively.

where $\mathbf{y}(t)$ collects measurements taken at time t by all robots associated with any landmark, and $\mathbf{M}$ and $\mathbf{R}$ are diagonal matrices, where their jth diagonal block is $\mathbf{M}_j$ and $\mathbf{R}_j$, respectively.[2]

Moreover, we assume that the robots are equipped with object recognition systems that allow them to collect measurements associated with the landmark classes $\mathbf{c}$. These semantic measurements typically consist of label measurements along with label probabilities [53]–[55]. Since such algorithms are scale and orientation invariant, the class measurement is independent of the robot/sensor position (assuming that the landmark is within the sensing range of the robot). Thus, once a landmark ℓ_i is detected, a semantic measurement is generated by the following observation model: $[\mathbf{y}_j^c, \mathbf{s}_j^c] = \mathbf{g}_j(L_i)$ (see also [56]), where $\mathbf{y}_j^c$ and $\mathbf{s}_j^c$ represent a class/label measurement and the corresponding probability scores over all available classes, respectively, and L_i stands for the true class of the detected landmark ℓ_i. More details about modeling sensors generating semantic measurements can be found in [56]; see also Remark 2.7. Hereafter, we compactly denote the object recognition model of all robots as

$$[\mathbf{y}^c(t), \mathbf{s}^c(t)] = \mathbf{g}(\mathbf{L}) \tag{4}$$

where $\mathbf{L}$ denotes the true classes of all landmarks detected at time t.

D. Separation Principle: Offline Update of Semantic Maps

Given measurements generated by the sensors (3) and (4) until a time instant $t \geq 0$, the semantic map can be updated online by fusing the collected measurements using recently proposed semantic SLAM algorithms [33], [34]. Note that updating the map requires knowledge of the measurements that the robots may collect in the field if they apply a sequence of control

[2]Note that the observation model (3) is assumed to be linear with respect to the landmark positions as this allows us to employ separation principles developed in estimation theory; see Section II-D. Nonlinear models (e.g., range sensors) are considered in Section IV.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:45:49 UTC from IEEE Xplore. Restrictions apply.

actions, denoted $\mathbf{u}_{0:t}$, which is impossible to obtain during the offline control synthesis process.

To mitigate this challenge and to design control policies while accounting for their effect on the map uncertainty, we rely on separation principles that have recently been employed in estimation theory [31], [32]. Specifically, given a linear observation model with respect to the landmark positions, as defined in (3), and given a sequence of control inputs until a time instant t, denoted by $\mathbf{u}_{0:t}$, the *a posteriori* covariance matrix can be computed offline using the Kalman filter Riccati equation, i.e., without requiring knowledge of any measurements [31], [32]. Hereafter, with slight abuse of notation, we denote the Kalman filter Riccati map by $\Sigma(t+1) = \rho(\Sigma(t), \mathbf{p}(t+1))$, where $\Sigma(t+1)$ and $\Sigma(t)$ denote the *a posteriori* covariance matrix at consecutive time instants t and $t+1$. Observe that $\Sigma(t+1)$ can be computed directly from $\Sigma(t)$ given the next robot position $\mathbf{p}(t+1)$ (or equivalently the control input $\mathbf{u}(t)$ applied at $\mathbf{p}(t)$). Note that the *a posteriori* estimates $\hat{\mathbf{x}}(t)$ for the positions of the landmarks cannot be updated offline, i.e., without sensor measurements generated by (3). Similarly, the semantic uncertainty, captured by the discrete distribution d, cannot be updated offline as it requires measurements (i.e., images) that will be processed by the object recognition algorithm (4).

Finally, throughout this article, we make the following assumption that is required for applying a Kalman filter and, consequently, for offline computation of *a posteriori* covariance matrices.

Assumption 2.3: The observation model (3) and the covariance matrix $\mathbf{R}$ of the measurement noise are known.

Assumption 2.3 is common in related works, and, in practice, it can be satisfied by learning the covariance matrix offline using training data [31]–[33].

E. Temporal Logic Tasks Over Uncertain Semantic Maps

The goal of the robots is to accomplish a complex collaborative task captured by a global co-safe LTL specification ϕ. To reason about both environmental and perceptual uncertainty, similar to [20] and [21], we define perception-based atomic predicates that depend on both the multi-robot state and the perceived uncertain semantic map over which the LTL formula is defined. To define such predicates, first, with slight abuse of notation, we denote by $\mathcal{M}(t)$ the map distribution at time t obtained after the robots apply a sequence of control actions $\mathbf{u}_{0:t}$; the offline construction of $\mathcal{M}(t)$ was discussed in Section II-D.

1) Perception-Based Predicates: We define perception-based predicates, denoted by $\pi_p(\mathbf{p}(t), \mathcal{M}(t), \Delta)$, as follows:

$$\pi_p(\mathbf{p}(t), \mathcal{M}(t), \Delta) = \begin{cases} \text{true}, & \text{if } p(\mathbf{p}(t), \mathcal{M}(t), \Delta) \geq 0 \\ \text{false}, & \text{otherwise} \end{cases} \tag{5}$$

where Δ is the set of user-specified and case-specific parameters (e.g., probabilistic thresholds or robots indices) and $p(\mathbf{p}(t), \mathcal{M}(t), \Delta) : \Omega^N \times \mathcal{M}(t) \times \Delta \to \mathbb{R}$. Hereafter, when it is clear from the context, we simply denote a perception-based predicate by π_p.

In what follows, we define functions $p(\mathbf{p}(t), \mathcal{M}(t), \Delta)$ that will be used throughout this article. First, we define the following two functions reasoning about the metric uncertainty of the map:

$$p(\mathbf{p}(t), \mathcal{M}(t), \{j, \ell_i, r, \delta\}) = \mathbb{P}(||\mathbf{p}_j(t) - \mathbf{x}_i|| \leq r) - (1-\delta) \tag{6}$$

and

$$p(\mathbf{p}(t), \mathcal{M}(t), \{j, \ell_i, \delta\}) = \delta - \det(\Sigma_i(t)). \tag{7}$$

The predicate associated with (6) is true at time t if the probability of robot j being within distance less than r from landmark ℓ_i is greater than $1-\delta$, after applying control actions $\mathbf{u}_{0:t}$, for some user-specified parameters $r, \delta > 0$. Similarly, (7) is true if the determinant of the *a posteriori* covariance matrix associated with landmark ℓ_i is less than a user-specified threshold δ after applying control actions $\mathbf{u}_{0:t}$. Note that in (7), robot j is responsible for actively localizing landmark ℓ_i, although the global multirobot state $\mathbf{p}(t)$ is used to compute the *a posteriori* covariance matrix $\Sigma(t)$ using the Kalman filter Riccati map. The latter predicate allows us to reason about how confident the robots are about the position of landmark ℓ_i; the smaller the $\det(\Sigma_i(t))$ is, the smaller the metric uncertainty about landmark i is.

In a similar way, we can define predicates that reason about both the metric and the semantic uncertainty. For instance, consider the following function:

$$p(\mathbf{p}(t), \mathcal{M}(t), \{j, r, \delta, c\}) = \max_{\ell_i} [\mathbb{P}(||\mathbf{p}_j(t) - \mathbf{x}_i|| \leq r) d_i(c)] - (1-\delta). \tag{8}$$

In words, the predicate associated with (8) is true at time t if the probability of robot j being within distance less than r from at least one landmark with class c is greater than $1-\delta$. Additional predicates can be defined using, e.g., the entropy of the discrete distributions or risk measures defined over the semantic map. For simplicity, in what follows, we restrict our attention on predicates of the form (6)–(8). We can also introduce atomic predicates defined over the robot states and the known geometric structure that have been widely used in the literature (e.g., atomic predicates that are true if a robot is within a known region of interest). For simplicity of presentation, we abstain from defining such atomic propositions. Finally, given a finite set of available parameters Δ, we define atomic predicates of the form (6)–(8) that are collected in a set $\mathcal{AP}$.

2) Syntax and Semantics: Given a set $\mathcal{AP}$ of perception-based atomic predicates, we construct a mission specification expressed in co-safe LTL, which is satisfied by a finite-length robot trajectory. The syntax of co-safe LTL over sequences of robot states and uncertain semantic maps is defined as follows.

Definition 2.4 (Syntax): The syntax of co-safe LTL over a sequence of multi-robot states $\mathbf{p}(t)$ and uncertain semantic maps $\mathcal{M}(t)$ is defined as $\phi ::= \text{true} | \pi_p | \neg \pi_p | \phi_1 \wedge \phi_2 | \phi_1 \vee \phi_2 | \phi_1 \mathcal{U} \phi_2$, where: 1) π_p is a perception-based predicate defined in (5) and 2) $\wedge$, $\vee$, $\neg$, and $\mathcal{U}$ denote the conjunction, disjunction, negation, and until operator, respectively.

Note that using the "until" operator, the eventually operator, denoted by $\Diamond$, can be defined as well [5].

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:45:49 UTC from IEEE Xplore. Restrictions apply.

Next, we define the semantics of co-safe LTL over sequences of multi-robot states and semantic maps. We first define a labeling function $L : \Omega^N \times \mathcal{M}(t) \to 2^{\mathcal{AP}}$ determining which atomic propositions are true given the current robot system state $\mathbf{p}(t) \in \Omega^N$ and the current map distribution $\mathcal{M}(t)$ obtained after applying a sequence of control actions $\mathbf{u}_{0:t}$.

Definition 2.5 (Semantics): Let $\sigma = \sigma(0), \sigma(1) \dots$ be a finite sequence of multirobot states and uncertain semantic maps, where $\sigma(t) = L([\mathbf{p}(t), \mathcal{M}(t)])$. The semantics of temporal logic over robot states and semantic maps is defined recursively as

$$\sigma \models \text{true}$$

$$\sigma \models \pi_p(\mathbf{p}, \mathcal{M}, \Delta) \Leftrightarrow p(\underbrace{\mathbf{p}(0), \mathcal{M}(0)}_{\text{from } \sigma(0)}, \Delta) \geq 0$$

$$\sigma \models \phi_1 \wedge \phi_2 \Leftrightarrow (\sigma \models \phi_1) \wedge (\sigma \models \phi_2)$$

$$\sigma \models \phi_1 \vee \phi_2 \Leftrightarrow (\sigma \models \phi_1) \vee (\sigma \models \phi_2)$$

$$\sigma \models \phi_1 \mathcal{U} \phi_2 \Leftrightarrow \exists t \geq 0 \text{s.t.} \{\sigma(t)\sigma(t+1) \dots \models \phi_2\}$$
$$\wedge \{\sigma(t')\sigma(t'+1) \dots \models \phi_1\} \quad \forall 0 \leq t' < t.$$

Example 2.6 (Mission specification): Consider a single-robot mission defined over a semantic map modeled as two landmarks with available classes $\mathcal{C} = \{\text{“Pole,” “Person”}\}$. The robot has to eventually reach a person of interest to deliver supplies while always avoiding colliding with a pole; see, e.g., Fig. 1. This task can be captured by the following co-safe LTL formula:

$$\begin{aligned} \phi = [\Diamond \pi_p(\mathbf{p}(t), \mathcal{M}(t), \{1, r_1, \delta_1, \text{“Person”}\})] \\ \wedge [\neg \pi_p(\mathbf{p}(t), \mathcal{M}(t), \{1, r_2, \delta_2, \text{“Pole”}\}) \\ \mathcal{U} \pi_p(\mathbf{p}(t), \mathcal{M}(t), \{1, r_1, \delta_1, \text{“Person”}\})] \end{aligned} \tag{9}$$

where the atomic predicates are defined as in (8).

F. Safe Planning Over Uncertain Semantic Maps

Given a task specification ϕ, an uncertain prior belief about the semantic map (see Assumption 2.1), the observation model (3), and the robot dynamics (2), our goal is to select a stopping horizon H and a sequence $\mathbf{u}_{0:H}$ of control inputs $\mathbf{u}(t)$, for all $t \in \{0, \dots, H\}$, that satisfy ϕ while minimizing a user-specified motion cost function. This gives rise to the following optimal control problem:

$$\min_{H, \mathbf{u}_{0:H}} \left[J(H, \mathbf{u}_{0:H}) = \sum_{t=0}^{H} c(\mathbf{p}(t), \mathbf{p}(t+1)) \right] \tag{10a}$$

$$[\mathbf{p}_{0:H}, \mathcal{M}_{0:H}] \models \phi \tag{10b}$$

$$\mathbf{p}(t) \in \Omega^N_{\text{free}} \tag{10c}$$

$$\mathbf{p}(t+1) = \mathbf{f}(\mathbf{p}(t), \mathbf{u}(t)) \tag{10d}$$

$$\hat{\mathbf{x}}(t+1) = \hat{\mathbf{x}}(0) \tag{10e}$$

$$\Sigma(t+1) = \rho(\Sigma(t), \mathbf{p}(t+1)) \tag{10f}$$

$$d(t+1) = d(0) \tag{10g}$$

where the constraints (10d)–(10g) hold for all time instants $t \in [0, H]$. In (10a), any motion cost function $c(\mathbf{p}(t), \mathbf{p}(t+1))$ can be used associated with the transition cost from $\mathbf{p}(t)$ to $\mathbf{p}(t+1))$ as long as it is positive (e.g., traveled distance). If non-positive summands are selected, then (10) is not well defined, since the optimal terminal horizon H is infinite. The constraints (10b) and (10c) require the robots to accomplish the mission specification ϕ and always avoid the known obstacles/walls, respectively. With slight abuse of notation, in (10b), $[\mathbf{p}_{0:H}, \mathcal{M}_{0:H}]$ denotes a finite sequence of length/horizon H of multirobot states and semantic maps, while $[\mathbf{p}_{0:H}, \mathcal{M}_{0:H}] \models \phi$ means that the symbols generated along this finite sequence satisfy ϕ (see also Definition 2.5). The constraint (10d) requires the robots to move according to their known dynamics (2). In addition, the constraints in (10e) and (10f) require the metric uncertainty to be updated as per the Kalman filter Riccati map, discussed in Section II-D, while the semantic uncertainty is not updated during the control synthesis process captured by (10g); instead, as will be discussed in Section III, it is updated online as semantic measurements are collected as per (4). Then, the problem addressed in this article can be summarized as follows.

Problem 1: Given an initial robot configuration $\mathbf{p}(0)$, a prior distribution of maps $\mathcal{M}(0)$ (see Assumption 2.1), a linear sensor model (3), and a task captured by a co-safe formula ϕ, compute a horizon H and compute control inputs $\mathbf{u}(t)$ for all time instants $t \in \{0, \dots, H\}$ as per (10).

In Section IV, we provide a discussion on relaxing the assumptions of 1) a linear observation model (3) and 2) availability of prior information about the metric/semantic environmental uncertainty discussed in Assumption 2.1.

Remark 2.7 (Online replanning and object recognition): The object recognition method modeled in (4) is not required to solve (10) as, therein, the semantic uncertainty is not updated. In fact, (10) is an offline problem yielding open-loop/offline robot paths that are agnostic to the object recognition model (4). A sampling-based algorithm to solve (10) is presented in Sections III-B and III-C. The online execution of these paths is discussed in Section III-D. Specifically, in that section, we discuss when the offline synthesized paths may need to be revised online to adapt to the semantic map that is continuously learned using perceptual feedback captured by (3) and (4); see Fig. 2 as well.

Remark 2.8 (Arbitrary LTL formulas): Recall that the optimal control problem (10) generates a finite sequence of control actions that satisfies a co-safe LTL specification. Arbitrary LTL formulas can be considered as well, which, however, are satisfied by infinite-length paths. Nevertheless, infinite-length paths cannot be manipulated in practice. Typically, to mitigate this issue, these paths are expressed in a prefix–suffix form, where the prefix path is executed once followed by the indefinite execution of the suffix path; see, e.g., [15]. Following this approach, to account for arbitrary LTL formulas, two separate optimal control problems, which have a similar form to (10), can be defined, where the first one generates a (finite) prefix path, while the second one generates the corresponding (finite) suffix path. Both optimal control problems can be solved by the proposed sampling-based

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:45:49 UTC from IEEE Xplore. Restrictions apply.

approach. In this article, for simplicity, we consider only co-safe LTL formulas that are satisfied by finite-horizon paths.

III. Safe Planning in Uncertain Semantic Maps

In this section, first, we show that (10) can be converted into a reachability problem defined over a hybrid and uncertain space; see Section III-A. Due to the hybrid nature of the state space that needs to be explored, (10) cannot be solved in a straightforward manner by existing reachability-based planning methods, such as [39] and [57]. To address this challenge, in Sections III-B and III-C, we present a sampling-based algorithm that can efficiently explore hybrid and uncertain state spaces and solve Problem 1. Then, in Section III-D, we discuss the online execution of the synthesized paths, and we present a replanning method to adapt to the continuously learned semantic map; see also Fig. 2.

A. Reachability in Uncertain Hybrid Spaces

To convert (10) into a reachability problem, we first convert the specification ϕ, defined over a set of atomic predicates $\mathcal{AP}$, into a deterministic finite state automaton (DFA), defined as follows [5].

Definition 3.1 (DFA): A DFA D over $\Sigma = 2^{\mathcal{AP}}$ is defined as a tuple $D = (\mathcal{Q}_D, q_D^0, \Sigma, \delta_D, q_F)$, where $\mathcal{Q}_D$ is the set of states, $q_D^0 \in \mathcal{Q}_D$ is the initial state, Σ is an alphabet, $\delta_D : \mathcal{Q}_D \times \Sigma \to \mathcal{Q}_D$ is a deterministic transition relation, and $q_F \in \mathcal{Q}_D$ is the accepting/final state.

Given a robot trajectory $\mathbf{p}_{0:H}$ and a corresponding sequence of maps $\mathcal{M}_{0:H}$, we get the labeled sequence $L(\mathbf{p}_{0:H}, \mathcal{M}_{0:H}) = L([\mathbf{p}(0), \mathcal{M}(0)]) \dots L([\mathbf{p}(H), \mathcal{M}(H)])$. This labeled sequence satisfies the specification ϕ, if starting from the initial state q_D^0, each symbol/element in $L(\mathbf{p}_{0:H}, \mathcal{M}_{0:H})$ yields a DFA transition so that eventually—after H DFA transitions—the final state q_F is reached [5]. As a result, we can equivalently rewrite (10) as follows:

$$\min_{H, \mathbf{u}_{0:H}} \left[J(H, \mathbf{u}_{0:H}) = \sum_{t=0}^{H} c(\mathbf{p}(t), \mathbf{p}(t+1)) \right] \tag{11a}$$

$$q_D(t+1) = \delta_D(q_D(t), \sigma(t)) \tag{11b}$$

$$\mathbf{p}(t+1) = \mathbf{f}(\mathbf{p}(t), \mathbf{u}(t)) \tag{11c}$$

$$\mathbf{p}(t) \in \Omega_{\text{free}}^N \tag{11d}$$

$$\hat{\mathbf{x}}(t+1) = \hat{\mathbf{x}}(0) \tag{11e}$$

$$\Sigma(t+1) = \rho(\Sigma(t), \mathbf{p}(t+1)) \tag{11f}$$

$$d(t+1) = d(0) \tag{11g}$$

$$q_D(H) = q_F \tag{11h}$$

where $q_D(0) = q_D^0$, $\sigma(t) = L([\mathbf{p}(t), \mathcal{M}(t)])$, and $\mathcal{M}(t)$ is determined by $\hat{\mathbf{x}}(t)$, $\Sigma(t)$, and $d(t)$. Note that the constraint in (11b) captures the automaton dynamics, i.e., the next DFA state that will be reached from the current DFA state under the observation/symbol $\sigma(t)$. In words, (11) is a reachability problem defined over a joint space consisting of the automaton state space [see (11b)], the robot motion space [see (11c) and (11d)], and the metric uncertainty space [see (11e)–(11g)], while the terminal constraint requires to reach the final automaton state [see (11h)].

B. Sampling-Based Planning in Uncertain Semantic Maps

In this section, we propose a sampling-based algorithm to solve (11), which incrementally builds trees that explore simultaneously the robot motion space, the space of covariance matrices (metric uncertainty), and the automaton space that corresponds to the assigned specification. The proposed algorithm is summarized in Algorithm 1.

In what follows, we provide some intuition for the steps of Algorithm 1. First, we denote the constructed tree by $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, J_{\mathcal{G}}\}$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of edges. The set of nodes $\mathcal{V}$ contains states of the form $\mathbf{q}(t) = [\mathbf{p}(t), \mathcal{M}(t), q_D(t)]$, where $\mathbf{p}(t) \in \Omega$ and $q_D(t) \in \mathcal{Q}_D$.[3] The function $J_{\mathcal{G}} : \mathcal{V} \to \mathbb{R}_+$ assigns the cost of reaching node $\mathbf{q} \in \mathcal{V}$ from the root of the tree. The root of the tree, denoted by $\mathbf{q}(0)$, is constructed so that it matches the initial robot state $\mathbf{p}(0)$, the initial semantic map $\mathcal{M}(0)$, and the initial DFA state, i.e., $\mathbf{q}(0) = [\mathbf{p}(0), \mathcal{M}(0), q_D^0]$. By convention, the cost of the root $\mathbf{q}(0)$ is $J_{\mathcal{G}}(\mathbf{q}(0)) = 0$, while the cost of a node $\mathbf{q}(t+1) \in \mathcal{V}$, given its parent node $\mathbf{q}(t) \in \mathcal{V}$, is computed as

$$J_{\mathcal{G}}(\mathbf{q}(t+1)) = J_{\mathcal{G}}(\mathbf{q}(t)) + c(\mathbf{p}(t), \mathbf{p}(t+1)). \tag{12}$$

Observe that by applying (12) recursively, we get that $J_{\mathcal{G}}(\mathbf{q}(t+1)) = J(t+1, \mathbf{u}_{0:t+1})$, which is the objective function in (10).

The tree $\mathcal{G}$ is initialized so that $\mathcal{V} = \{\mathbf{q}(0)\}$, $\mathcal{E} = \emptyset$, and $J_{\mathcal{G}}(\mathbf{q}(0)) = 0$ (line 2, Algorithm 1). In addition, the tree is built incrementally by adding new states $\mathbf{q}_{\text{new}}$ to $\mathcal{V}$ and corresponding edges to $\mathcal{E}$, at every iteration n of Algorithm 1, based on a *sampling* (lines 4–6, Algorithm 1) and *extending-the-tree* operation (lines 7–21, Algorithm 1). After taking $n_{\max} \geq 0$ samples, where $n_{\max}$ is user specified, Algorithm 1 terminates and returns a feasible solution to (11) (if it has been found), i.e., a terminal horizon H and a sequence of control inputs $\mathbf{u}_{0:H}$.

To extract such a solution, we need first to define the set $\mathcal{X}_g \subseteq \mathcal{V}$ that collects all states $\mathbf{q}(t) = [\mathbf{p}(t), \mathcal{M}(t), q_D(t)] \in \mathcal{V}$ of the tree that satisfy $q_D(t) = q_F$, which captures the terminal constraint (11h) (lines 19 and 20, Algorithm 1). Among all nodes in $\mathcal{X}_g$, we select the node $\mathbf{q}(t) \in \mathcal{X}_g$, with the smallest cost $J_{\mathcal{G}}(\mathbf{q}(t))$, denoted by $\mathbf{q}(t_{\text{end}})$ (line 22, Algorithm 1). Then, the terminal horizon is $H = t_{\text{end}}$, and the control inputs $\mathbf{u}_{0:H}$ are recovered by computing the path $\mathbf{q}_{0:t_{\text{end}}}$ in $\mathcal{G}$ that connects $\mathbf{q}(t_{\text{end}})$ to the root $\mathbf{q}(0)$, i.e., $\mathbf{q}_{0:t_{\text{end}}} = \mathbf{q}(0), \dots, \mathbf{q}(t_{\text{end}})$ (line 23, Algorithm 1); see also Fig. 2. Note that satisfaction of the constraints in (11) is guaranteed by construction of $\mathcal{G}$; see Section III-C. In what follows, we describe the core operations of Algorithm 1, "*sample*" and "*extend*" that are used to construct the tree $\mathcal{G}$.

[3]Throughout this article, when it is clear from the context, we drop the dependence of $\mathbf{q}(t)$ on t.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:45:49 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1: Mission Planning in Probabilistic Maps.

Input: (i) maximum number of iterations $n_{\max}$, (ii) dynamics (2), (iii) map distribution $\mathcal{M}(0)$, (iv) initial robot configuration $\mathbf{p}(0)$, (v) task ϕ;

Output: Terminal horizon H, and control inputs $\mathbf{u}_{0:H}$

1 Convert ϕ into a DFA;
2 Initialize $\mathcal{V}=\{\mathbf{q}(0)\}$, $\mathcal{E}=\emptyset$, $\mathcal{V}_1=\{\mathbf{q}(0)\}$, $K_1=1$, and $\mathcal{X}_g=\emptyset$;
3 **for** $n=1,\ldots,n_{max}$ **do**
4 Sample a subset $\mathcal{V}_{k_{\text{rand}}}$ from $f_{\mathcal{V}}$;
5 **for** $\mathbf{q}_{rand}(t)=[\mathbf{p}_{rand}(t),\mathcal{M}_{rand}(t),q_D]\in\mathcal{V}_{k_{rand}}$ **do**
6 Sample a control input $\mathbf{u}_{\text{new}}\in\mathcal{U}$ from $f_{\mathcal{U}}$;
7 $\mathbf{p}_{\text{new}}(t+1)=\mathbf{f}(\mathbf{p}_{\text{rand}}(t),\mathbf{u}_{\text{new}})$;
8 **if** $\mathbf{p}_{new}(t+1)\in\Omega^N_{free}$ **then**
9 $\hat{\mathbf{x}}_{\text{new}}(t+1)=\hat{\mathbf{x}}(0)$;
10 $\Sigma_{\text{new}}(t+1)=\rho(\Sigma_{\text{rand}}(t),\mathbf{p}_{\text{new}}(t+1))$;
11 $d_{\text{new}}(t+1)=d(0)$;
12 Construct map: $\mathcal{M}_{\text{new}}(t+1)=(\hat{\mathbf{x}}_{\text{new}}(t+1),\Sigma_{\text{new}}(t+1),d_{\text{new}}(t+1))$;
13 Compute $q_D^{\text{new}}=\delta_D(q_D^{\text{rand}},L([\mathbf{p}_{\text{rand}}(t),\mathcal{M}_{\text{rand}}(t)]))$;
14 **if** $\exists q_D^{new}$ **then**
15 Construct $\mathbf{q}_{\text{new}}=[\mathbf{p}_{\text{new}},\mathcal{M}_{\text{new}},q_D^{\text{new}}]$;
16 Update set of nodes: $\mathcal{V}=\mathcal{V}\cup\{\mathbf{q}_{\text{new}}\}$;
17 Update set of edges: $\mathcal{E}=\mathcal{E}\cup\{(\mathbf{q}_{\text{rand}},\mathbf{q}_{\text{new}})\}$;
18 Compute cost of new state: $J_{\mathcal{G}}(\mathbf{q}_{\text{new}})=J_{\mathcal{G}}(\mathbf{q}_{\text{rand}})+c(\mathbf{p}_{\text{rand}},\mathbf{p}_{\text{new}})$;
19 **if** $q_D^{new}=q_F$ **then**
20 $\mathcal{X}_g=\mathcal{X}_g\cup\{\mathbf{q}_{\text{new}}\}$;
21 Update the sets $\mathcal{V}_k$;
22 Among all nodes in $\mathcal{X}_g$, find $\mathbf{q}_{\text{end}}(t_{\text{end}})$;
23 $H=t_{\text{end}}$ and recover $\mathbf{u}_{0:H}$ by computing the path $\mathbf{q}_{0:t_{\text{end}}}=\mathbf{q}(0),\ldots,\mathbf{q}(t_{\text{end}})$;

C. Incremental Construction of Trees

At every iteration n of Algorithm 1, a new state $\mathbf{q}_{\text{new}}(t+1)=[\mathbf{p}_{\text{new}},\mathcal{M}_{\text{new}},q_D^{\text{new}}]$ is sampled. The construction of the state $\mathbf{q}_{\text{new}}(t+1)$ relies on three steps. Specifically, first, we sample a state $\mathbf{p}_{\text{new}}$; see Section III-C1. Second, given $\mathbf{p}_{\text{new}}$ and its parent node, we compute the corresponding new semantic map $\mathcal{M}_{\text{new}}$. Third, we append to $\mathbf{p}_{\text{new}}$ and $\mathcal{M}_{\text{new}}$ a DFA state q_D^{new} giving rise to $\mathbf{q}_{\text{new}}(t+1)$, which is then added to the tree structure, if possible; see Section III-C2.

1) Sampling Strategy: To construct the state $\mathbf{p}_{\text{new}}$, we first divide the set of nodes $\mathcal{V}$ into a *finite* number of sets, denoted by $\mathcal{V}_k\subseteq\mathcal{V}$, based on the robot state $\mathbf{p}$ and the DFA state q_D that comprise the states $\mathbf{q}\in\mathcal{V}$. Specifically, $\mathcal{V}_k$ collects all states $\mathbf{q}\in\mathcal{V}$ that share the same DFA state and the same robot state (or in practice, robot states that very close to each other). By construction of $\mathcal{V}_k$, we get that $\mathcal{V}=\cup_{k=1}^{K_n}\{\mathcal{V}_k\}$, where K_n is the number of subsets $\mathcal{V}_k$ at iteration n. In addition, notice that K_n is finite for all iterations n, due to the finite number of available control inputs $\mathbf{u}$ and the finite DFA state-space. At iteration $n=1$ of Algorithm 1, it holds that $K_1=1$ and $\mathcal{V}_1=\mathcal{V}$ (line 2, Algorithm 1).

Second, given the sets $\mathcal{V}_k$, we first sample from a given discrete distribution $f_{\mathcal{V}}(k|\mathcal{V}):\{1,\ldots,K_n\}\to[0,1]$ an index $k\in\{1,\ldots,K_n\}$ that points to the set $\mathcal{V}_k$ (line 4, Algorithm 1). The mass function $f_{\mathcal{V}}(k|\mathcal{V})$ defines the probability of selecting the set $\mathcal{V}_k$ at iteration n of Algorithm 1 given the set $\mathcal{V}$.

Next, given the set $\mathcal{V}_{k_{\text{rand}}}$ sampled from $f_{\mathcal{V}}$, we perform the following steps for all $\mathbf{q}\in\mathcal{V}_{k_{\text{rand}}}$. Specifically, given a state $\mathbf{q}_{\text{rand}}$, we sample a control input $\mathbf{u}_{\text{new}}\in\mathcal{U}$ from a discrete distribution $f_{\mathcal{U}}(\mathbf{u}):\mathcal{U}\to[0,1]$ (line 6, Algorithm 1). Given a control input $\mathbf{u}_{\text{new}}$ sampled from $f_{\mathcal{U}}$, we construct the state $\mathbf{p}_{\text{new}}$ as $\mathbf{p}_{\text{new}}=\mathbf{f}(\mathbf{p}_{\text{rand}},\mathbf{u}_{\text{new}})$ (line 7, Algorithm 1). If $\mathbf{p}_{\text{new}}$ belongs to the obstacle-free space, as required by (11d), then the "extend" operation follows (line 8, Algorithm 1).

Any mass functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ can be used to generate $\mathbf{p}_{\text{new}}$ as long as they satisfy the following two assumptions. These assumptions will be used in Section V to show completeness and optimality of the proposed sampling-based algorithm.

Assumption 3.2 (Probability mass function $f_{\mathcal{V}}$): (i) The probability mass function $f_{\mathcal{V}}(k|\mathcal{V}):\{1,\ldots,K_n\}\to[0,1]$ satisfies $f_{\mathcal{V}}(k|\mathcal{V})\geq\epsilon\ \forall\ k\in\{1,\ldots,K_n\}$ and for all $n\geq 0$, for some $\epsilon>0$ that remains constant across all iterations n. (ii) Independent samples k_{rand} can be drawn from $f_{\mathcal{V}}$.

Assumption 3.3 (Probability mass function $f_{\mathcal{U}}$): (i) The distribution $f_{\mathcal{U}}(\mathbf{u})$ satisfies $f_{\mathcal{U}}(\mathbf{u})\geq\zeta$, for all $\mathbf{u}\in\mathcal{U}$, for some $\zeta>0$ that remains constant across all iterations n. (ii) Independent samples $\mathbf{u}_{\text{new}}$ can be drawn from the probability mass function $f_{\mathcal{U}}$.

Remark 3.4 (Mass functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$): Assumptions 3.2(i) and 3.3(i) also imply that the mass functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ are bounded away from zero on $\{1,\ldots,K_n\}$ and $\mathcal{U}$, respectively. In addition, observe that Assumptions 3.2 and 3.3 are very flexible, since they also allow $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ to change with iterations n of Algorithm 1, as the tree grows.

Remark 3.5 (Sampling strategy): An example of a distribution $f_{\mathcal{V}}$ that satisfies Assumption 3.2 is the discrete uniform distribution $f_{\mathcal{V}}(k|\mathcal{V})=\frac{1}{k}$, for all $k\in\{1,\ldots,K_n\}$. Observe that the uniform mass function trivially satisfies Assumption 3.2(ii). In addition, observe that Assumption 3.2(i) is also satisfied, since there exists a finite $\epsilon>0$ that satisfies Assumption 3.2(i), which is $\epsilon=\frac{1}{|\mathcal{R}|}$, where $\mathcal{R}$ is a set that collects all robot configurations $\mathbf{p}$ that can be reached by the initial state $\mathbf{p}(0)$. Note that $\mathcal{R}$ is a finite set, since the set $\mathcal{U}$ of admissible control inputs is finite, by assumption. Similarly, uniform mass functions $f_{\mathcal{U}}$ satisfy Assumption 3.3. Note that any functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ can be employed as long as they satisfy Assumptions 3.2 and 3.3. Nevertheless, the selection of $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ affects the performance of Algorithm 1; see the numerical experiments in Section VI. In Appendix B, we design (nonuniform) mass functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ that bias the exploration toward locations that are expected to contribute to the satisfaction of the assigned mission specification.

2) Extending the Tree: To build incrementally a tree that explores both the robot motion space, the space of metric uncertainty, and the DFA state space, we need to append to

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:45:49 UTC from IEEE Xplore. Restrictions apply.

$\mathbf{p}_{\text{new}}$ the corresponding semantic map $\mathcal{M}_{\text{new}}$ determined by the parameters $(\hat{\mathbf{x}}_{\text{new}}, \Sigma_{\text{new}}, d_{\text{new}})$ and DFA state q_D^{new}. Particularly, $\mathcal{M}_{\text{new}}$ is constructed so that $\hat{\mathbf{x}}_{\text{new}} = \hat{\mathbf{x}}(0)$, as required in (11e), and computing Σ_{new} as per the Kalman filter Riccati map, i.e., $\Sigma_{\text{new}} = \rho(\Sigma_{\text{rand}}, \mathbf{p}_{\text{new}})$, as required in (11f), and $d_{\text{new}} = d(0)$, as required in (11g) (lines 9–12, Algorithm 1). Next, to construct the state $\mathbf{q}_{\text{new}}$ we append to $\mathbf{p}_{\text{new}}$ and $\mathcal{M}_{\text{new}}$ the DFA state $q_D^{\text{new}} = \delta_D(q_D^{\text{rand}}, L([\mathbf{p}_{\text{rand}}, \mathcal{M}_{\text{rand}}]))$, as required by (11b). In words, q_D^{new} is the automaton state that that can be reached from the parent automaton state q_D^{rand} given the observation $L([\mathbf{p}_{\text{rand}}, \mathcal{M}_{\text{rand}}])$. If such a DFA state does not exist, then this means that the observation $L([\mathbf{p}_{\text{rand}}, \mathcal{M}_{\text{rand}}])$ results in violating the LTL formula, and this new sample is rejected. Otherwise, the state $\mathbf{q}_{\text{new}} = (\mathbf{p}_{\text{new}}, \mathcal{M}_{\text{new}}, q_D^{\text{new}})$ is constructed (line 15, Algorithm 1), which is then added to the tree.

Given a state $\mathbf{q}_{\text{new}}$, we update the set of nodes and edges of the tree as $\mathcal{V} = \mathcal{V} \cup \{\mathbf{q}_{\text{new}}\}$ and $\mathcal{E} = \mathcal{E} \cup \{(\mathbf{q}_{\text{rand}}, \mathbf{q}_{\text{new}})\}$, respectively (lines 16 and 17, Algorithm 1). The cost of the new node $\mathbf{q}_{\text{new}}$ is computed as in (12), i.e., $J_{\mathcal{G}}(\mathbf{q}_{\text{new}}) = J_{\mathcal{G}}(\mathbf{q}_{\text{rand}}) + c(\mathbf{p}_{\text{rand}}, \mathbf{p}_{\text{new}})$ (line 18, Algorithm 1). Finally, the sets $\mathcal{V}_k$ are updated, so that if there already exists a subset $\mathcal{V}_k$ associated with both the DFA state $\mathbf{q}_D^{\text{new}}$ and the robot state $\mathbf{p}_{\text{new}}$, then $\mathcal{V}_k = \mathcal{V}_k \cup \{\mathbf{q}_{\text{new}}\}$. Otherwise, a new set $\mathcal{V}_k$ is created, i.e., $K_n = K_n + 1$ and $\mathcal{V}_{K_n} = \{\mathbf{q}_{\text{new}}\}$ (line 21, Algorithm 1). Recall that this process is repeated for all states $\mathbf{q}_{\text{rand}} \in \mathcal{V}_{k_{\text{rand}}}$ (line 5, Algorithm 1).

Remark 3.6 (Finite decomposition of $\mathcal{V}$): In Section III-C1, we decomposed the set of nodes $\mathcal{V}$ into K_n finite sets, where K_n remains finite as $n \to \infty$, by construction. The *finite* decomposition of the set of nodes is required to ensure probabilistic completeness and optimality of the proposed sampling-based algorithm; see Section V. Any alternative decomposition of $\mathcal{V}$ that always results in a finite number of subsets $\mathcal{V}_k$ can be employed without affecting correctness of Algorithm 1; see also Section VI-A.

D. Online Execution and Replanning

Algorithm 1 generates an open-loop sequence $\mathbf{q}_{0:H} = \mathbf{q}(0), \mathbf{q}(1), \ldots, \mathbf{q}(H)$, where $\mathbf{q}(t) = [\mathbf{p}(t), \mathcal{M}(t), q_D(t)]$, so that the resulting robot trajectory $\mathbf{p}_{0:H}$ and sequence of maps $\mathcal{M}_{0:H}$ satisfy ϕ. As the robots execute synchronously their respective paths in $\mathbf{p}_{0:H}$, synthesized by Algorithm 1, they take measurements [see (3) and (4)], which are used to update both the metric and the semantic environmental uncertainty using a semantic SLAM method. In other words, every time the robots move toward their next waypoint, they take measurements, which are then used to update: 1) the *a posteriori* mean and covariance matrix associated with the landmark positions and 2) the discrete distribution associated with the landmark classes yielding a new semantic map denoted by $\mathcal{M}_{\text{online}}(t)$. Note that the online constructed map $\mathcal{M}_{\text{online}}(t)$ may be different from the corresponding offline map $\mathcal{M}(t)$, as the latter is constructed by predicting only the metric environmental uncertainty by computing the *a posteriori* covariance matrices.

As a result, given the new map, the previously designed paths may not be feasible, and therefore, replanning may be required to adapt to the updated map $\mathcal{M}_{\text{online}}(t)$; see also Fig. 2. Specifically, replanning is needed only if the observations/symbols that the robots generate online do not allow them to follow the sequence of DFA states that appears in $\mathbf{q}_{0:H}$. Formally, replanning at time t is required only if the observation $L([\mathbf{p}(t), \mathcal{M}_{\text{online}}(t)])$ does not enable a transition from $q_D(t)$ to $q_D(t+1)$, where $q_D(t)$ and $q_D(t+1)$ are determined by $\mathbf{q}_{0:H}$ computed by Algorithm 1. In this case, Algorithm 1 is initialized with the current robot state $\mathbf{p}(t)$, the current DFA state $q_D(t)$, and the current map $\mathcal{M}_{\text{online}}(t)$ to generate a new path. This process is repeated until the accepting DFA state is reached. Once this happens, the LTL formula has been satisfied, and the mission terminates. We note that it is possible that at time t, as the robots follow their paths, the LTL formula may get violated. This happens if $L([\mathbf{p}(t), \mathcal{M}_{\text{online}}(t)])$ does not yield any DFA transition from the current DFA state $q_D(t)$ (or if it yields a transition to an absorbing DFA state, in case DFA is complete). In this scenario, least violating paths can be found as, e.g., in [58].[4]

Remark 3.7 (Prior map and replanning): First, recall from Assumption 2.1 that we assume that there exist a known number of landmarks in the environment. This assumption can be relaxed due to the SLAM capability included in the proposed architecture; see Fig. 2. In fact, a prior semantic map is used to initialize Algorithm 1 and to guide the planning phase. Once the robots follow the designed paths, they may discover/sense new landmarks. Once this happens, the employed SLAM method will generate a probabilistic belief about these landmarks, which will then be included in the semantic map. Second, it is possible that the prior distribution may guide the robots toward locations in the environment where no landmarks exist. For instance, consider a case where, as per the prior distribution, a landmark of interest is expected to be in a certain room. Due to this prior distribution, Algorithm 1 will design paths toward this room. If no landmarks exist in this room, i.e., if this prior distribution is wrong, then it can be discarded from the current semantic map, so that design of mission paths is not guided by this wrong prior. For instance, a prior belief about a landmark can be considered wrong, if the robot approaches the expected position of the landmark within some distance $\eta > 0$, and no landmarks are sensed. The parameter η is user specified, and it can be associated with the sensing range of the robots. Both scenarios may trigger replanning as discussed above to account for the new map.

Remark 3.8 (Replanning frequency): The frequency with which replanning is triggered depends on how different the offline predicted map is from the online learned map. In Section VI, we illustrate through simulations that the more inaccurate the initial semantic map is, the more often replanning is triggered. A potential approach toward minimizing the replanning frequency

[4]Intuitively, violation of an LTL formula as the robots follow their paths depends on their sensing capabilities. For instance, an LTL formula may get violated during execution of the synthesized paths, if the robot sensing range is significantly smaller than the proximity requirements captured by the atomic predicates. Consider, e.g., the specification in Example 2.6; see also Fig. 1. Assume that a robot has to always keep a distance of 2 m from the pole. If the sensing capabilities allow the robot to detect objects that are within $d < 2$ m, then it is possible that the pole avoidance requirement may get violated, as the pole may get detected only when its distance from the robots is less than $d < 2$ m.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:45:49 UTC from IEEE Xplore. Restrictions apply.

is to incorporate semantic map prediction methods into the offline planning process [59] or to design paths that are robust to uncertainty [60], [61]. In addition, throughout this article, we have assumed that the geometric structure of the environment is known; note that the geometry of the environment can be learned using SLAM methods as well [48]. This assumption can be relaxed by replanning every time the robots detect new obstacles/walls that interfere with the paths designed by Algorithm 1. Nevertheless, this may result in rather frequent replanning, especially, in cluttered environments, increasing the online computational burden [62]. Our future work will focus on designing more efficient replanning frameworks in the presence of both geometric and metric/semantic environmental uncertainty by, e.g., locally patching the designed paths, or decomposing the global LTL task into smaller subtasks [47], or incorporating data-driven map prediction methods in the planning process [63].

IV. Discussion and Extensions

In this section, we discuss how assumptions made in Section II can be relaxed. Specifically, in Section IV-A, we discuss how the robots should behave in case feasible—either initial or online revised—paths do not exist. This way, we can relax Assumption 2.1 requiring an informative enough prior belief so that there exists a feasible solution to (10). Then, in Section IV-B, we show how Algorithm 1 can account for sensor models (3) that may be nonlinear in the positions of landmarks (e.g., range sensors) or subject to noise of unknown distribution.

A. Reactive Planning in Unexplored Semantic Environments

In this section, we discuss how the robots should behave in case feasible paths do not exist, and therefore, Algorithm 1 cannot generate paths. In fact, initial feasible paths may not exist if a prior map is unavailable or highly uncertain; see Assumption 2.1. The latter may hold for the online learned map when replanning is triggered as well. To mitigate this challenge, we couple Algorithm 1 with existing exploration and information gathering methods [32], [64], [65]. The key idea is that the robots switch back and forth between an exploration and exploitation mode depending on the mission status. In particular, first, the robots follow an exploration strategy aiming to spread in the environment to detect landmarks and decrease the environmental uncertainty. Once a user-specified condition about the exploration mode is satisfied, the robots switch to an exploitation mode by applying Algorithm 1 aiming to find paths satisfying the assigned mission by exploiting the semantic map constructed during exploration. If such paths still do not exist, the robots switch back to the exploration mode. This process is repeated until the robots accomplish their mission.

In what follows, we describe an exploration strategy, adopted from [32] and [64], along with its interaction with Algorithm 1 in more detail. First, we define a grid $C = \{c_1, c_2, \ldots, c_L\}$ over the environment, where c_k corresponds to the kth grid cell, and we split the grid cells into "explored" (i.e., cells that have been visited by the robots) and "unexplored." Then, we introduce dummy "exploration" landmarks at the frontiers of the explored grid map for which uniform discrete and Gaussian distributions are defined for their labels and positions, respectively. Given the "exploration" and any already detected landmarks, the robots design informative open-loop/offline paths that aim to minimize the metric uncertainty of these landmarks within a predetermined finite horizon $F > 0$; more details about designing these paths can be found in [32]. Note that the fake uncertainty in the exploration landmark locations promises information gain to the robots motivating them to move toward the unexplored part of the environment and, therefore, detect new landmarks. As the robots start following the designed paths while taking measurements and updating the semantic map, they may detect new landmarks. Once a new landmark ℓ_i is detected by a robot j, a Gaussian and a discrete distribution associated with this landmark are initialized. Once all robots have finished the execution of their paths, they should determine whether they should keep exploring the environment or design mission-based paths over the constructed semantic map using Algorithm 1. We require mission-based planning to be triggered once $K(t) \geq 0$ new landmarks are detected, where $K(t)$ is user specified and it depends on the assigned mission. In addition, additional requirements can be incorporated to trigger Algorithm 1 related to, e.g., the map uncertainty. Note that during the exploration phase, as the robots detect and localize landmarks, they may satisfy certain atomic propositions enabling a transition from the current DFA state to a new DFA state; see also Remark 4.1. Thus, when Algorithm 1 is executed, it is initialized with the DFA state that the exploration mode has ended up.

Remark 4.1 (Safe exploration): A particular challenge that arises in the proposed framework is that of safe exploration. In fact, as the robots explore the environment to detect and localize previously unknown landmarks they may satisfy atomic propositions that violate the assigned LTL specification. Specifically, let $q_D(t)$ be the automaton state the robots are at time t, and assume that the robots are currently in exploration mode. During exploration, the robots may generate a symbol that does not enable any transition originating from $q_D(t)$ resulting in violation of the LTL task. An empirical approach toward mitigating this issue is to ensure that the robots always satisfy the Boolean formula that enables them to remain in $q_D(t)$.[5] For instance, assume that the robots can remain in $q_D(t)$ if with probability greater than 0.9, they keep a distance of at least 1 m from any landmark with class "table." Then, we can compute ϵ-confidence intervals, for some $\epsilon > 0$, centered at the expected positions of detected landmarks whose most likely class is "table"; ϵ-confidence intervals are then treated as virtual obstacles that should be avoided. If the negation operator is excluded from the syntax, then exploration is always safe as there are no observations that can violate the assigned LTL task.

B. Nonlinear Sensor Models and Unknown Noise Distribution

In Section II, we assumed that the sensor model (3) is linear with respect to the landmark positions $\mathbf{x}$ allowing us to compute

[5]By construction of the DFA, there exists a Boolean formula defined over $\mathcal{AP}$ that needs to be satisfied to enable a DFA transition.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:45:49 UTC from IEEE Xplore. Restrictions apply.

offline the *a posteriori* covariance matrix without the need of measurements. Nevertheless, this assumption may not hold in practice, e.g., in case of range sensors. In this case, during the execution of Algorithm 1, similar to [32], we compute the *a posteriori* covariance matrices Σ_{new} using the linearized observation model about the estimated landmark positions (line 10, Algorithm 1). Such case studies are presented in Section VI.

Additionally, in Section II, we assumed that the measurement noise follows a Gaussian distribution. This can be relaxed by leveraging recent distributionally robust planning methods [66], [67]. Specifically, we can consider measurement noise with unknown distribution assuming that it belongs to an ambiguity set with known mean and covariance matrix. Then, the Kalman filter Riccati map can be used to update the covariance matrices offline [67]. A key challenge in this case is to define perception-based predicates capturing probabilistic requirements since the hidden state (e.g., the landmark positions) does not follow a known distribution. To mitigate this challenge, worst-case probabilistic requirements can be defined that are typically translated to linear constraints [67].

V. Completeness and Optimality

In this section, we examine the completeness and optimality of Algorithm 1. First, in Theorem 5.1, we show that Algorithm 1 is probabilistically complete. In other words, given an initial semantic map $\mathcal{M}(0)$ and a sensor model as per (3), if there exists a solution to (10), then Algorithm 1 will eventually find it. In Theorem 5.2, we also show that Algorithm 1 is asymptotically optimal, i.e., Algorithm 1 will eventually find the optimal solution to (10), if it exists. The proofs can be found in Appendix A.

To show completeness and optimality of Algorithm 1, we assume that the probability mass functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ satisfy Assumptions 3.2 and 3.3, respectively.

Theorem 5.1 (Probabilistic completeness): If there exists a solution to (10), then Algorithm 1 is probabilistically complete, i.e., the probability of finding a feasible solution, i.e., a feasible horizon H and a feasible sequence of control inputs $\mathbf{u}_{0:H}$ for (10), goes to 1 as $n \to \infty$.

Theorem 5.2 (Asymptotic optimality): Assume that there exists an optimal solution to (10). Then, Algorithm 1 is asymptotically optimal, i.e., the optimal horizon H and the optimal sequence of control inputs $\mathbf{u}_{0:H}$ will be found with probability 1, as $n \to \infty$. In other words, the path generated by Algorithm 1 satisfies $\mathbb{P}(\{\lim_{n\to\infty} J(H, \mathbf{u}_{0:H}) = J^*\}) = 1$, where J is the objective function of (10) and J^* is the optimal cost.[6]

Recall from Section III-D that the synthesized paths may need to be revised online by rerunning Algorithm 1. Theorems 5.1 and 5.2 imply that if replanning is required at any time t, a feasible and the optimal path will eventually be found, if they exist. Nevertheless, the proposed control architecture cannot guarantee recursive feasibility, i.e., existence of feasible paths at future time instants when replanning is needed. In fact, existence of feasible paths when replanning is triggered depends on the initialization of the corresponding optimal control problem (10), i.e., the online constructed map and, consequently, on the perceptual feedback, which is hard to predict *a priori*.

[6]Note that the horizon H and $\mathbf{u}_{0:H}$ returned by Algorithm 1 depend on n. For simplicity of notation, we drop this dependence.

TABLE I
Scalability Analysis: $\tau = 0.1$

N	M	Matlab Runtimes (mins) (First feasible path)		H	
		$\tau = 0.1$	$\tau = 0.5$	$\tau = 0.1$	$\tau = 0.5$
1	15	0.66	0.12	461	105
1	30	0.88	0.21	465	107
5	15	1.94	0.48	389	94
5	30	2.68	0.58	389	102
20	15	8.74	2.33	561	132
20	30	10.32	2.65	565	121
40	15	14.68	7.77	457	151
40	30	16.43	7.81	456	151
100	15	33.32	15.41	533	147
100	30	38.42	20.56	543	146

TABLE II
Scalability Analysis: $\tau = 0.5$

N	M	MatLab Runtime	H
10	10	0.94 mins	106
10	45	2.98 mins	146
10	60	3.94 mins	121
10	75	5.41 mins	133

VI. Experimental Validation

In this section, we present experimental case studies that illustrate the performance of Algorithm 1 and show that it can solve large-scale planning tasks in uncertain environments. In Section VI-A, we demonstrate scalability of Algorithm 1 with respect to the number of landmarks and robots. In particular, given an initial semantic map and an initial multirobot state, we report the time required by Algorithm 1 to find the first feasible solution along with the corresponding terminal horizon (without considering replanning). The scalability analysis has been conducted using MATLAB 2016b on a computer with Intel Core i7 3.1 GHz and 16-GB RAM. In Section VI-B, we demonstrate in a Gazebo simulated semantic city the replanning framework, discussed in Section III-D, for various initial semantic maps. These experiments have been conducted on Gazebo (ROS, python3) on a computer with Intel Core i5-8350U 1.7 GHz and 16-GB RAM. Finally, in Section VI-E, we show how Algorithm 1 can be applied in environments with no prior information about their metric/semantic structure, as per the discussion provided in Section IV. The experimental videos can be found in [68].

A. Scalability Analysis: Offline Planning Over Prior Semantic Maps

In what follows, we examine the scalability performance of Algorithm 1 with respect to the number N of robots, and the number M of landmarks. We first describe the robot dynamics, the sensor model, the setup of the initial semantic map, and the structure of the assigned LTL task, and then, we conclude with the scalability analysis summarized in Tables I and II.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:45:49 UTC from IEEE Xplore. Restrictions apply.

1) Robot Dynamics: We consider robots with the following differential drive dynamics (13), shown at bottom of this page, where the robot state $\mathbf{p}_j(t) = [p_j^1(t), p_j^2(t), \theta_j]^T$ captures both the position $[p_j^1(t), p_j^2(t)]$ and the orientation $\theta_j(t)$ of robot j and τ is the sampling period. The available motion primitives are $u \in \{0, 1\}$ m/s and $\omega \in \{0, \pm 1, \pm 2, \ldots, \pm 180\}^\circ$/s.

2) Sensor Model (3): All robots are equipped with omnidirectional, range-only, line-of-sight sensors with limited range of 1 m. Every robot can take noisy range measurements, i.e., measure its distance from all landmarks that lie within its sight and range. Specifically, the measurement associated with robot j and landmark i is given by

$$y_{j,i} = \ell_{j,i}(t) + v(t) \text{ if } (\ell_{j,i}(t) \leq R_j) \wedge (i \in \texttt{FOV}_j) \quad (14)$$

where $\ell_{j,i}(t)$ is the distance between landmark ℓ_i and robot j, R_j denotes the sensing range of robot j which is equal to 1 m, $\texttt{FOV}_j$ denotes its corresponding field of view, and $v(t) \sim \mathcal{N}(0, \sigma(\mathbf{x}_i(t), \mathbf{p}_j(t)))$ is the measurement noise. In addition, we model the measurement noise so that σ increases linearly with $\ell_{j,i}(t)$, with slope 0.5, as long as $\ell_{j,i}(t) \leq R_j$; if $\ell_{j,i}(t) > R_j$, then σ is infinite. This model captures the fact that range readings become more inaccurate as the distance increases [69]; nevertheless, any other model can be used, such as uniform noise within the sensing range. Observe that the considered observation model is nonlinear, and therefore, the *a posteriori* covariance matrix cannot be estimated optimally offline; see (11f). In this case, at every iteration of Algorithm 1, we approximately compute the *a posteriori* covariance matrices Σ_{new} using the linearized observation model about the predicted positions of the landmarks (line 10, Algorithm 1).[7]

3) Sampling Strategy: Hereafter, we employ the mass functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ designed in Appendix B with $p_{\text{rand}} = 0.9$ and $p_{\mathcal{U}} = 0.9$. Note that uniform sampling functions failed to solve the considered planning problems within 3 h due to the large joint state space that needs to be explored. In fact, using uniform sampling strategies, Algorithm 1 was able to solve only small-scale problems, e.g., single-agent planning problems in 4×4 environments with three landmarks. In addition, notice that the computational complexity of Algorithm 1 per iteration depends linearly on the size of the selected subset $\mathcal{V}_{k_{\text{rand}}}$ (see [line 5, Algorithm 1]). Therefore, to speed up the construction of a feasible solution, a new set $\mathcal{V}_k$ is constructed for every new sample, for the first 100 iterations/samples (i.e., the first 100 sets $\mathcal{V}_k$ are singleton). After that, the sets $\mathcal{V}_k$ are constructed, as described in Section III. This way, the number of subsets $\mathcal{V}_k$ remains finite as Algorithm 1; see also Remark 3.6.

[7]We avoid defining an object recognition method as per (4), as the main focus of this section is to demonstrate scalability of Algorithm 1 in terms of solving (10) without considering replanning. Recall that (10) does not depend on the object recognition method. In fact, the latter is needed only for learning online the semantic map and accordingly replanning; see Section III-D.

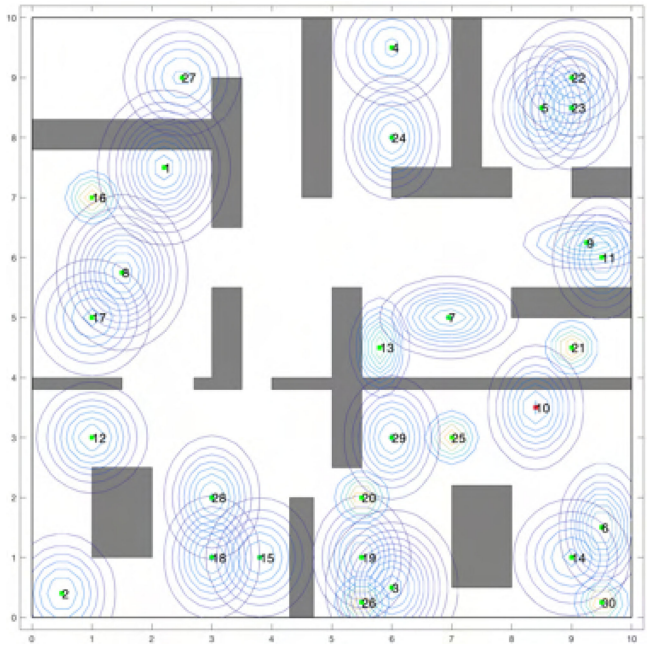


Fig. 4. Initial semantic map $\mathcal{M}(0)$ with 30 landmarks considered in Table I. The map with the 15 landmarks considered in the same table consists of the landmarks ℓ_i with indices $i \in \{1, \ldots, 15\}$ shown in this figure.

4) Initial Semantic Map: We have evaluated the performance of Algorithm 1 on two different initial semantic maps consisting of $M = 15$ and $M = 30$ landmarks illustrated in Fig. 4. Particularly, this figure shows the Gaussian distributions determining the initial semantic map $\mathcal{M}(0)$ along with the known geometric structure. We assume that the set of classes is $\mathcal{C} = \{1, 2, \ldots, M\}$, while the true class of each landmark is uncertain. In fact, the robots have access to a discrete distribution $d(0)$ over the labels that is designed by randomly selecting a probability within the range [0.8,1] that a landmark has the correct label, and by randomly selecting probabilities for the remaining classes.

5) Mission Specification: The robots are responsible for accomplishing a search-and-rescue task captured by the following LTL formula: $\phi = \Diamond\xi_1 \wedge \Diamond\xi_2 \wedge \neg\xi_1 \mathcal{U} \xi_3 \wedge \Diamond(\xi_4 \wedge \Diamond(\xi_5 \wedge$

$$\begin{bmatrix} p_j^1(t+1) \\ p_j^2(t+1) \\ \theta_j(t+1) \end{bmatrix} = \begin{bmatrix} p_j^1(t) \\ p_j^2(t) \\ \theta_j(t) \end{bmatrix} + \begin{cases} \begin{bmatrix} \tau u \cos(\theta_j(t) + \tau\omega/2) \\ \tau u \sin(\theta_j(t) + \tau\omega/2) \\ \tau\omega \end{bmatrix}, & \text{if } \tau\omega < 0.001 \\ \begin{bmatrix} \frac{u}{\omega}(\sin(\theta_j(t) + \tau\omega) - \sin(\theta_j(t))) \\ \frac{u}{\omega}(\cos(\theta_j(t)) - \cos(\theta_j(t) + \tau\omega)) \\ \tau\omega \end{bmatrix} & \text{else} \end{cases} \quad (13)$$

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:45:49 UTC from IEEE Xplore. Restrictions apply.

$(\Diamond\xi_6))) \wedge \Diamond\xi_7$ where ξ_i is a Boolean formula. Specifically, to define ξ_i, we decompose the robot team into smaller subgroups, where each subgroup is associated with a specific landmark. Each ξ_i is associated with a finite number of pairs of robot subgroups and an assigned landmark. We collect such pairs in a set denoted by $\mathcal{P}_i$. The Boolean formula ξ_i is true if the following is true for all pairs in $\mathcal{P}_i$: given a pair in $\mathcal{P}_i$, all robots within a subgroup should be within $r = 0.2$ m from a specific landmark ℓ_k with the probability greater than $1 - \delta = 0.75$ [see (6)], while the uncertainty about the position of this landmark is less than $\delta = 0.01$ [see (7)]. In words, the specification ϕ requires: (i) subgroups of robots to eventually satisfy ξ_1 (see $\Diamond\xi_1$); (ii) subtask (i) to never happen until certain subgroups satisfy ξ_3 [see $(\neg\xi_1 \mathcal{U} \xi_3)$]; (iii) robot subgroups to eventually satisfy ξ_2 (see $\Diamond\xi_2$); (iii) robot subgroups to eventually satisfy ξ_7 [see $(\Diamond\xi_7)$]; and (iv) robot subgroups to eventually satisfy ξ_4, ξ_5, and ξ_6 in this order [see $\Diamond(\xi_4 \wedge \Diamond(\xi_5 \wedge (\Diamond\xi_6)))$]. Subtasks (i)–(iv) should be satisfied according to the probabilistic satisfaction requirements discussed above.

Observe that the LTL task ϕ does not capture the requirement for obstacle avoidance. Recall that obstacle avoidance is considered during the offline path design in (line 8, Algorithm 1). Additional safety constraints can be added to line 8. For instance, consider a safety property that requires all robots to always avoid approaching landmark with class 10 within distance of $r = 1$ m with the probability greater than $1 - \delta = 0.2$. This requirement can be captured by the following Boolean formula $\xi_{\text{safety}} = \wedge_{j\in\mathcal{N}} \neg\pi_p(\mathbf{p}(t), \mathcal{M}(t), \{j, 1, 0.8, \text{“10”}\})$, where the atomic predicate π_p is defined as per (8). Given such a safety property, every time a new sample $\mathbf{p}_{\text{new}}$ is generated, we can compute the corresponding semantic map $\mathcal{M}_{\text{new}}$ and check if the current pair of the multirobot state $\mathbf{p}_{\text{new}}$ and the semantic map $\mathcal{M}_{\text{new}}$ satisfies the Boolean formula ξ_{safety}. If it does, the state $\mathbf{q}_{\text{new}}$ can be constructed as discussed in Algorithm 1. Otherwise, the current sample is discarded. In this case study, we execute Algorithm 1 for the specification ϕ along with the additional safety constraint ξ_{safety} defined above.

6) Scalability Results: In what follows, we examine the performance of Algorithm 1 with respect to the number N of robots, and the number M of landmarks along with the effect of robot velocities [parameter τ in (13)]. In this scalability analysis, for a fixed number M of landmarks, as we increase the number of robots, we associate each ξ_i in ϕ with more pairs of robot subteams and landmarks. The same holds as we increase the number of landmarks given a fixed number N of robots as well. In addition, we note that the mission specification does not necessarily involve all landmarks that may exist in the environment. The results are summarized in Table I for $\tau = 0.1$ and $\tau = 0.5$ s. First, using available online toolboxes [70], we convert ϕ into a deterministic automaton that has 48 states and 540 transitions. The pruning process finished in less than a second for all cases studies considered in Table I, which, e.g., for the case study $N = 100, M = 30$, yielded a pruned automaton with 172 edges. Observe in Table I that as the robot velocities increase, the terminal horizon and the runtime of Algorithm 1 decrease. In addition, observe in Table II that for a fixed number of number of robots, as the number of landmarks increase, the total runtime of Algorithm 1 increases, as the robots have to adapt planning to a “larger” map. Notice that Algorithm 1 can solve large-scale tasks that may involve hundreds of robots in a reasonable time due to the biased sampling process. Recall that the employed biased sampling strategy requires computing the geodesic distance between the position of robot j to the expected position of a landmark, for all robots j. In our implementation, to compute the geodesic distances, we first discretize the workspace, and then, we use the Dijkstra algorithm to compute the corresponding shortest paths sequentially across the robots. As a result, the runtimes can significantly decrease if the geodesic distances are computed in parallel.

B. Effect of Prior Map Uncertainty on Replanning Frequency

In this section, we present experimental studies that involve AsTech Firefly unmanned aerial vehicles (UAVs) that operate over a semantic city with dimensions 150×150. There are $M = 5$ landmarks in the city, and the available classes are “pole” and “person.”

1) Robot Dynamics: The AsTech Firefly UAV is governed by first-order dynamics, where the UAV state includes the position, velocity, orientation, and biases in the measured angular velocities and acceleration; more details can be found in [71]. In general, the more complex the robot dynamics is, the longer it takes for sampling-based methods to generate feasible paths as they have to explore a larger state and control space. To mitigate this issue, an approach that we investigate in this section is to generate paths for simple robot dynamics that need to be followed by robots with more complex dynamics. Specifically, in what follows, we use Algorithm 1 to synthesize paths considering the differential drive dynamics defined in (13) that are simpler than the actual AsTech Firefly UAV dynamics. Given the waypoints, generated by Algorithm 1, we compute minimum snap trajectories that smoothly transition through all waypoints every $T = 2$ s [72]. The UAVs are controlled to follow the synthesized trajectories using the ROS package developed in [71].

2) Perception System (3), (4): We assume that the drones are equipped with a downward facing sensor with rectangular field of view with dimensions 24×16 m that can take noisy positional measurements of landmarks falling inside its field of view (see, e.g., Fig. 5 and [52]), i.e., the measurement of landmark ℓ_i by robot j is

$$\mathbf{y}_{j,i} = \mathbf{x}_j + \mathbf{v} \tag{15}$$

where $\mathbf{v}$ is the Gaussian noise with a covariance matrix with diagonal entries equal to 2.

As for the object recognition method (4), we assume that the robot is equipped with a neural network that is capable of detecting classes of interest, which are, for this case study, “car,” “pole,” and “person” and returning a discrete distribution over the available classes (see, e.g., [55]). In addition, we model the confusion matrix of the object recognition method as follows:

$$\begin{pmatrix} 0.8 & 0.23 & 0.06 \\ 0.18 & 0.75 & 0.04 \\ 0.02 & 0.02 & 0.9 \end{pmatrix}$$

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:45:49 UTC from IEEE Xplore. Restrictions apply.

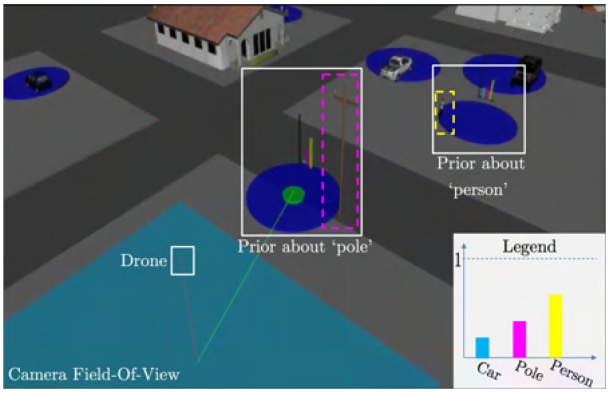


Fig. 5. A drone is responsible for delivering supplies to a landmark with class "person" while always avoiding hitting landmarks with class "pole" as per (9). The prior information about two landmarks is also illustrated. Observe that the prior discrete distribution for the "pole" is inaccurate in the sense that, initially, the most likely class of this landmark is "person."

where the first, second, and third row and column correspond to the classes "person," "car," and "pole," respectively. The entries in this confusion matrix capture the conditional probabilities $\mathbb{P}(\text{predicted class (row)}|\text{actual class (column)})$. For instance, if the drone sees a person, then with probability 0.8, 0.18, and 0.02, it will generate a discrete distribution, where the most likely class is "person," "car," and "pole," respectively. The confusion matrix can be learned offline based on training data. We note that any other object recognition/classification method can be used in place of the simulated one as this is used only to update the environmental belief.

3) Mission Specifications and Reacting to the Learned Map: First, we consider a single drone that is responsible for accomplishing the task (9) described in Example 2.6 with parameters $r_1 = r_2 = 0.2$ m, $\delta_1 = 0.2$, and $\delta_2 = 0.8$. The prior semantic map that is available to the drone is illustrated in Fig. 5. As per the prior Gaussian distribution, it holds that $||\hat{\mathbf{x}}_1(0) - \mathbf{x}_i|| = 4.9$ m and $||\hat{\mathbf{x}}_2(0) - \mathbf{x}_i|| = 5$ m, where ℓ_1 and ℓ_2 refer to the landmarks with true classes "person" and "pole," respectively. In addition, the semantic prior is inaccurate as the most likely class for landmark ℓ_2, i.e., the pole (that has to be avoided), is "person" (who has to be approached). In addition, the most likely class for landmark ℓ_1, i.e., the person, is "person." Due to this (partially) wrong semantic prior, the drone initially heads toward the "pole" [see Fig. 6(a)]. As the drone navigates the environment, it updates the semantic map, and it accordingly replans so that eventually the person is approached and the pole is avoided as per the imposed probabilistic requirements [see Fig. 6(b)–(f)].[8] In total, replanning was triggered five times. Snapshots of the robot trajectory along with instances where replanning was triggered are provided and discussed in Fig. 6. Additional simulations for various initial semantic maps can be found in [68], where it can

[8]In our implementation, when replanning is triggered, a new tree is constructed from scratch. A more efficient approach would be to exploit the previously constructed tree structure. For instance, given the new online learned map, the *a posteriori* covariance matrices for all tree nodes that are reachable from the current one (where replanning was triggered) can be computed along with the corresponding DFA states. Then, replanning may start using this subtree. This way, previously sampled obstacle-free multirobot states are not discarded.

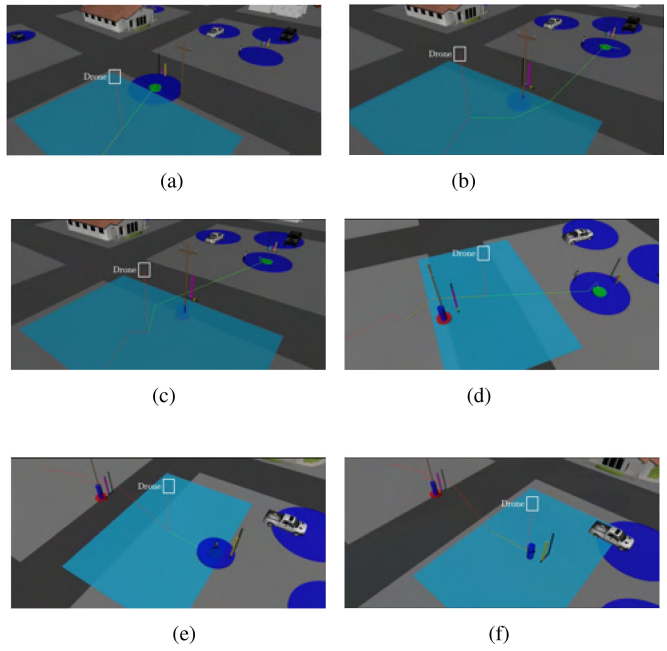


Fig. 6. Snapshots of a drone navigating an environment toward accomplishing the task in (9). The green solid line shows the path the drone currently plans to follow. The drone initially heads toward the "pole" [see (a)] believing that this landmark is a "person" due to its inaccurate prior information (see also Fig. 5). Once the "pole" is approached, the robot updates its belief about this landmark and it replans so that the "pole" is avoided while heading to another landmark that is (correctly) believed to be a "person" [see (b) and (c)]. Once the person is detected, the belief about the position and the class of this landmark is updated, the drone replans and moves toward the person [see (d) and (e)]. Once the person is approached as per the imposed probabilistic requirements, the mission terminates [see (f)]. (a) Heading toward the "pole." (b) and (c) Replanning. (d) Avoiding "pole" and heading toward "person." (e) Heading toward the "person." (f) Mission accomplished.

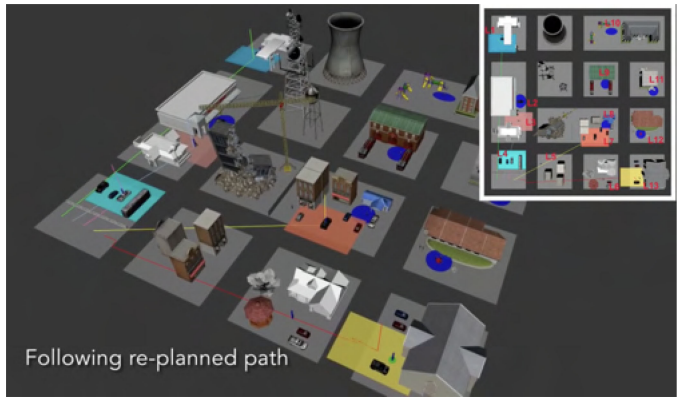
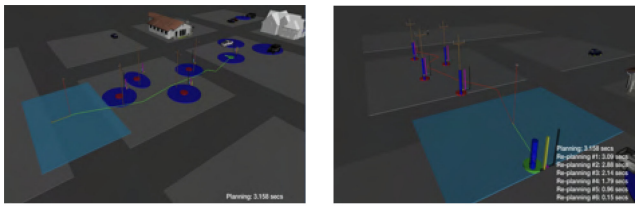


Fig. 7. A team of drones equipped with downward facing cameras (colored rectangles) is responsible for accomplishing a collaborative mission.

be observed that the more inaccurate the initial semantic map is, the more often replanning is triggered. For instance, replanning was never triggered when the prior means of the two landmarks (i.e., person and pole) were close enough to the actual positions, i.e., $||\hat{\mathbf{x}}_1(0) - \mathbf{x}_1|| = 0.7$ m and $||\hat{\mathbf{x}}_2(0) - \mathbf{x}_2|| = 1$ m, and the most likely class, as per the prior distribution $d(0)$, for each landmark, is the true one. Similar observations were made for case studies involving a team of five drones; see Fig. 7 and [68].

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:45:49 UTC from IEEE Xplore. Restrictions apply.



(a) (b)

Fig. 8. Drone tasked with reaching a person while avoiding four poles. (a) Initial map. (b) Approaching the person.

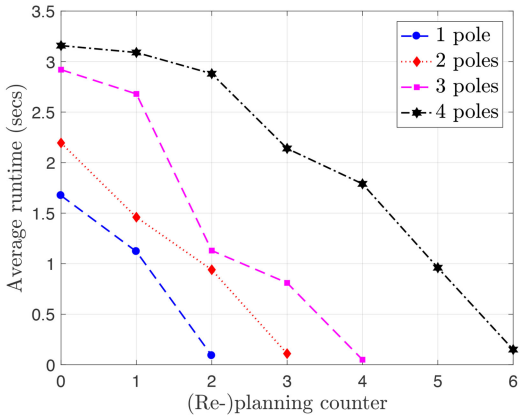


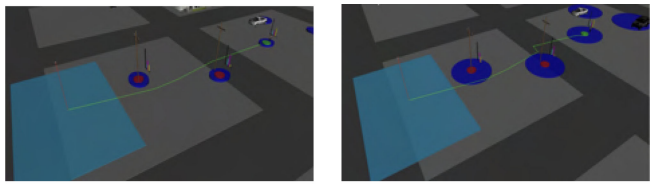
Fig. 9. Graphical depiction of replanning frequency and average (re)planning runtime. The runtime when the replanning counter is zero refers to the runtime to design the initial paths. Each curve corresponds to a different number of poles. The markers along the curves correspond to replanning events. The average total planning runtime for one, two, three, and four poles is 0.961, 1.176, 1.518, and 2.024 s. Note that these average runtimes are significantly smaller (between 0.1 and 0.3 s) when Algorithm 1 is not executed in the Gazebo simulation environment.

C. Effect of Map Density on Replanning Frequency

In this section, we build upon the case study considered in Section VI-B, and we examine how the density of uncertain obstacles/poles affects the replanning frequency and runtimes. Specifically, we consider a single drone with the dynamics, perceptual capabilities, and the mission specification discussed in Section VI-B. Recall that the assigned mission (9) requires the drone to eventually reach a person while always avoiding all poles with a user-specified probability; see also Fig. 8. In Fig. 9, we report the average (re)planning runtimes as the number of the poles increases. Specifically, to design the initial or revised paths, we run Algorithm 1 five times, and we report the average runtime. Observe in this figure that as the number of uncertain poles increases, the replanning frequency increases as well. Additionally, notice that as the mission progresses (i.e., as the replanning counter increases), the runtimes tend to decrease, as the robot gets "closer" to accomplishing the assigned task.

D. Benefit of Predicting Offline the Metric Uncertainty

In this section, we compare the proposed method against [30]. Recall that Algorithm 1 updates the covariance matrices of the semantic map as per (11f) as opposed to [30], where the map



(a) (b)

Fig. 10. Semantic maps of an environment with two poles and a person. (a) Semantic map with an informative metric prior distribution. (b) Less informative map; observe the flatter Gaussian distributions.

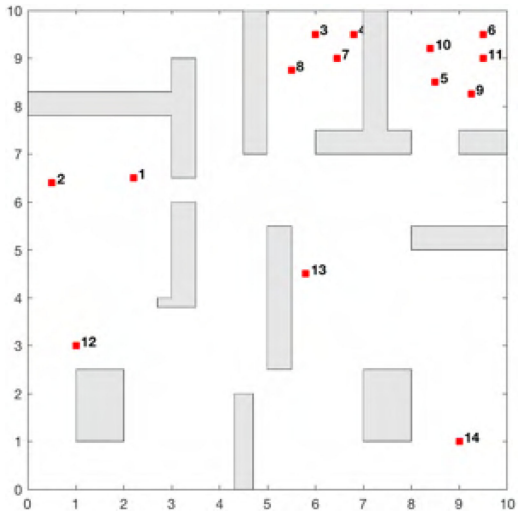


Fig. 11. Graphical depiction of the environment considered in Section VI-E. The gray polygons represent known obstacles, while the red squares denote the true positions of the landmarks. The numbers next to the landmarks denote their respective indices. Each index correspond to a different label, i.e., there are $|\mathcal{C}| = 14$ available classes. For instance, 14 (bottom right corner) corresponds to "resource center," while 9 (top right corner) corresponds to "injured person." The robot does not know either the positions or the classes of the landmarks.

remains fixed during the synthesis phase. Particularly, we consider the case study discussed in Section VI-B evaluated on two semantic maps shown in Fig. 10(a) and (b). The only difference between these two maps is that the prior Gaussian distributions in Fig. 10(a) are more informative (or less uncertain) than the ones in Fig. 10(b). When the informative map in Fig. 10(a) is considered, both [30] and Algorithm 1 generate feasible initial paths after 1.152 and 1.548 s, with costs 47 m and 46 m and terminal horizons for both equal to 13, respectively, while replanning was triggered three times for both algorithms. When the less informative map in Fig. 10(b) is considered, Algorithm 1 found a feasible path after 4.824 s. To the contrary, [30] failed to design feasible initial paths as the imposed probabilistic requirements cannot be satisfied under the considered prior metric uncertainty. In other words, due to updating offline the covariance matrices, Algorithm 1 can find feasible paths in prior maps with high metric uncertainty.

E. Safe Planning in Semantically Unexplored Environments

In this section, we evaluate Algorithm 1 in environments with known geometric structure but with no prior semantic map, i.e.,

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:45:49 UTC from IEEE Xplore. Restrictions apply.

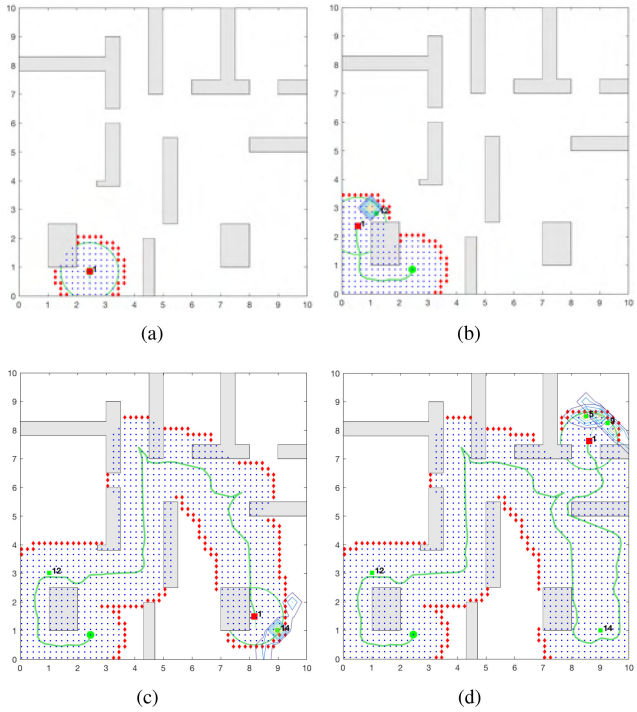


Fig. 12. Exploration mode. (a)–(d) illustrate the robot path toward exploring the environment. The green disk, red square, and the green solid line represent the initial robot location, the current robot location, and the followed trajectory, respectively. The green circle centered at the current robot position denotes the sensing range. The green squares denote the actual position of the detected landmarks. Gaussian distributions for landmarks ℓ_i with $\det \Sigma_i < 10^{-5}$ are not shown. The numbers next to the landmarks show the most likely class. The blue dots represent the cells of the grid map that have been explored, while the red diamonds show the "fake" exploration landmarks. At time instants $t = 40$, $t = 225$, and $t = 335$, the landmarks ℓ_{12} and ℓ_{14} ("resource center") and ℓ_5 and ℓ_9 ("injured person") are detected, respectively. Mission-based planning (see Algorithm 1) was triggered at the time instants $t = 225$, and $t = 335$. Feasible paths were found only at $t = 335$ illustrated in Fig. 13. (a) $t = 0$. (b) $t = 40$. (c) $t = 225$. (d) $t = 335$.

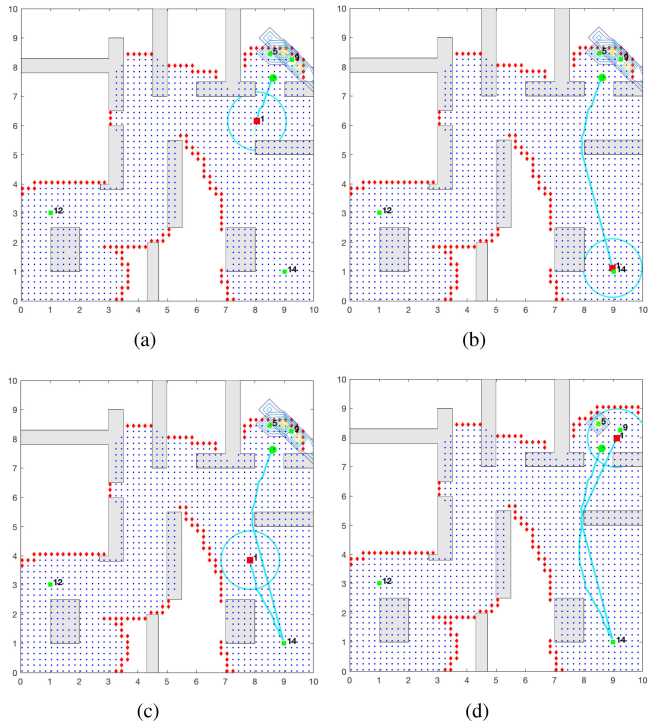


Fig. 13. Exploitation mode. (a)–(d) illustrate the robot path (cyan solid line) to accomplish the task (16) designed at time $t = 335$ (green disk) when Algorithm 1 successfully designed feasible paths. From $t = 335$ until $t = 380$, the robot moves toward landmark ℓ_{14} ("resource center"). Once this landmark is visited as per the imposed probabilistic requirement, the robot heads toward landmark ℓ_9 ("injured person"), which is visited at $t = 435$, and then, the mission terminates. (a) $t = 350$. (b) $t = 380$. (c) $t = 410$. (d) $t = 435$.

when Assumption 2.1 does not hold. Specifically, in this case, the number M of landmarks is unknown, while a prior uncertain semantic map $\mathcal{M}(0)$ is not available. In this case, Algorithm 1 is applied, as discussed in Section IV.

In what follows, we consider a single ground robot with the dynamics defined in (13) equipped with: 1) a camera capable of taking noisy positional measurements [as in (15)] of all landmarks that are within range of 1 m, where the covariance matrix of the Gaussian measurement noise has diagonal entries equal to 0.3 and 2) an object recognition method, as defined in Section VI-B2. The robot operates in a 10×10 m environment with the semantic and geometric structure presented in Fig. 11. The robot is responsible for eventually finding a landmark with class "resource center" to pick up supplies and, then, eventually find an "injured person," where the collected supplies will be delivered, while always avoiding known obstacles. Formally, this sequencing mission can be expressed by the following LTL formula:

$$\phi = \Diamond\{\pi_p(\mathbf{p}(t), \mathcal{M}(t), \{1, 0.2, 0.1, \text{“resource center”}\}) \wedge \Diamond[\pi_p(\mathbf{p}(t), \mathcal{M}(t), \{1, 0.2, 0.1, \text{“injured person”}\})]\} \quad (16)$$

where $\pi_p(\mathbf{p}(t), \mathcal{M}(t), \{1, 0.2, 0.1, \text{“resource center”}\})$ and $\pi_p(\mathbf{p}(t), \mathcal{M}(t), \{1, 0.2, 0.1, \text{“injured person”}\})$ are atomic predicates defined as per (8). For instance, the latter requires the robot, with the probability greater than 0.9 to be within a distance of 0.2 m from a landmark with class "injured person." In this case study, we assume that landmarks ℓ_9 and ℓ_{14} are the landmarks with classes "injured person" and "resource center," respectively; see also Fig. 11.

We emphasize that, initially, the robot does not have any prior information about where the landmarks of interest are. As a result, the robot initially adopts an exploration mode aiming to detect landmarks of interest; see Fig. 12(a). Here, since the mission involves two distinct landmarks, we require the robot to switch from exploration to exploitation (i.e., application of Algorithm 1 over the currently learned semantic map) for the first time when the first two landmarks are detected. This happens at the time instant $t = 225$ [see Fig. 12(c)]. Algorithm 1 fails to design a path satisfying the assigned task based on the currently learned map within 1000 iterations. Thus, the robot keeps being in exploration mode and switches to exploitation mode every time a new landmark is detected. At $t = 335$, both landmarks ℓ_9 and ℓ_5 are detected, and then, Algorithm 1 is triggered generating feasible paths over the currently learned semantic map. Then, the robot switches to an exploitation mode following the synthesized paths; see Fig. 13. Observe in Fig. 13 that the robot does not explore the whole environment, but, instead, it

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:45:49 UTC from IEEE Xplore. Restrictions apply.

only partially learns/explores the environment to accomplish the assigned mission. This simulation required in total 51.2 s.

VII. Conclusion

In this article, we addressed a multi-robot mission planning problem in environments with partially unknown semantics. In particular, the semantic environment was modeled as static labeled landmarks with uncertain positions and classes. We considered robots equipped with noisy perception systems that are tasked with accomplishing collaborative tasks with probabilistic requirements captured by co-safe LTL. We proposed a sampling-based algorithm to design open-loop sensor-based control policies that can accomplish the assigned mission. The synthesized policies are updated online to adapt to the semantic map that is continuously learned using semantic SLAM methods. Theoretical and experimental analyses were provided to support the proposed method. Our future work will focus on designing: 1) distributionally robust motion planning algorithms to account for unmodeled sensor noise and 2) fast reactive planning frameworks that can quickly adapt to metric/semantic uncertainty and the potentially dynamic geometric structure of the environment.

Appendix A
Proof of Completeness, Optimality, and Complexity

In what follows, we denote by $\mathcal{G}^n = \{\mathcal{V}^n, \mathcal{E}^n, \mathcal{J}_\mathcal{G}\}$ the tree that has been built by Algorithm 1 by the nth iteration. The same notation also extends to $f_\mathcal{V}$, $f_\mathcal{U}$, and $\mathbf{u}_{\text{new}}$. To prove Theorems 5.1 and 5.2, it suffices to show that Algorithm 1 can find all possible paths $\mathbf{q}_{0:H}$, for any $H > 0$, whether they are feasible or not, with probability 1 as $n \to \infty$. Showing this result means that if there exists a feasible (including the optimal) path $\mathbf{q}_{0:H}$, then Algorithm 1 will find it with probability 1 as $n \to \infty$. The key idea is to prove that once any node is added to the tree, then, eventually, this node will be extended toward all possible feasible directions as per the set of control inputs $\mathcal{U}$. A key assumption to show this is that of finite control space $\mathcal{U}$. To show this result, we first need to state the following results.

Lemma A.1 (Sampling $\mathcal{V}^n_{k_{rand}}$): Consider any subset $\mathcal{V}^n_k$ and any fixed iteration index n and any fixed $k \in \{1, \ldots, K_n\}$. Then, there exists an infinite number of subsequent iterations $n + w$, where $w \in \mathcal{W} \subseteq \mathbb{N}$ and $\mathcal{W}$ is a subsequence of $\mathbb{N}$, at which the subset $\mathcal{V}^n_k$ is selected to be the set $\mathcal{V}^{n+w}_{k_{\text{rand}}}$.

Proof: Let $A^{\text{rand},n+w}(k) = \{\mathcal{V}^{n+w}_{k_{\text{rand}}} = \mathcal{V}^n_k\}$, with $w \in \mathbb{N}$, denote the event that at iteration $n + w$ of Algorithm 1, the subset $\mathcal{V}^n_k \subseteq \mathcal{V}^n$ is selected by the sampling operation to be the set $\mathcal{V}^{n+w}_{k_{\text{rand}}}$ (line 4, Algorithm 1). In addition, let $\mathbb{P}(A^{\text{rand},n+w}(k))$ denote the probability of this event, i.e., $\mathbb{P}(A^{\text{rand},n+w}(k)) = f^{n+w}_\mathcal{V}(k)$.

Next, define the infinite sequence of events $A^{\text{rand}} = \{A^{\text{rand},n+w}(k)\}^\infty_{w=0}$, for a given subset $\mathcal{V}^n_k \subseteq \mathcal{V}^n$. In what follows, we show that the series $\sum^\infty_{w=0} \mathbb{P}(A^{\text{rand},n+w}(k))$ diverges, and then, we complete the proof by applying the Borel–Cantelli lemma [73].

Recall that by Assumption 3.2(i), we have that given any subset $\mathcal{V}^n_k \subseteq \mathcal{V}^n$, the probability $f^n_\mathcal{V}(k|\mathcal{V}^n)$ satisfies $f^n_\mathcal{V}(k|\mathcal{V}^n) \geq \epsilon$, for any iteration n. Thus, we have that $\mathbb{P}(A^{\text{rand},n+w}(k)) = f^{n+w}_\mathcal{V}(k|\mathcal{V}^{n+w}) \geq \epsilon > 0$, for all $w \in \mathbb{N}$. Note that this result holds for any $k \in \{1, \ldots, K_{n+w}\}$ due to Assumption 3.2(i). Therefore, we have that $\sum^\infty_{w=0} \mathbb{P}(A^{\text{rand},n+w}(k)) \geq \sum^\infty_{w=0} \epsilon$. Since ϵ is a strictly positive constant, we have that $\sum^\infty_{w=0} \epsilon$ diverges. Then, we conclude that $\sum^\infty_{w=0} \mathbb{P}(A^{\text{rand},n+w}(k)) = \infty$. Combining this result with the fact that the events $A^{\text{rand},n+w}(k)$ are independent by Assumption 3.2(ii), we get that $\mathbb{P}(\limsup_{k\to\infty} A^{\text{rand},n+w}(k)) = 1$, by the Borel–Cantelli lemma. In other words, the events $A^{\text{rand},n+w}(k)$ occur infinitely often, for all $k \in \{1, \ldots, K_n\}$. This equivalently means that for every subset $\mathcal{V}^n_k \subseteq \mathcal{V}^n$, for all $n \in \mathbb{N}_+$, there exists an infinite subsequence $\mathcal{W} \subseteq \mathbb{N}$ so that for all $w \in \mathcal{W}$, it holds $\mathcal{V}^{n+w}_{k_{\text{rand}}} = \mathcal{V}^n$, completing the proof. □

Lemma A.2 (Sampling $\mathbf{u}_{new}$): Consider any subset $\mathcal{V}^n_{k_{\text{rand}}}$ selected by $f_\mathcal{V}$ and any fixed iteration index n. Then, for any given control input $\mathbf{u} \in \mathcal{U}$, there exists an infinite number of subsequent iterations $n + w$, where $w \in \mathcal{W}'$ and $\mathcal{W}' \subseteq \mathcal{W}$ is a subsequence of the sequence of $\mathcal{W}$ defined in Lemma 1.1, at which the control input $\mathbf{u} \in \mathcal{U}$ is selected to be $\mathbf{u}^{n+w}_{\text{new}}$.

Proof: Define the infinite sequence of events $A^{\text{new}} = \{A^{\text{new},n+w}(\mathbf{u})\}^\infty_{w=0}$, for $\mathbf{u} \in \mathcal{U}$, where $A^{\text{new},n+w}(\mathbf{u}) = \{\mathbf{u}^{n+w}_{\text{new}} = \mathbf{u}\}$, for $w \in \mathbb{N}$, denotes the event that at iteration $n + w$ of Algorithm 1 the control input $\mathbf{u} \in \mathcal{U}$ is selected by the sampling function to be the input $\mathbf{u}^{n+w}_{\text{new}}$, given the subset $\mathcal{V}^n_{k_{\text{rand}}} \in \mathcal{V}^{n+w}_{k_{\text{rand}}}$. Moreover, let $\mathbb{P}(A^{\text{new},n+w}(\mathbf{u}))$ denote the probability of this event, i.e., $\mathbb{P}(A^{\text{new},n+w}(\mathbf{u})) = f^{n+w}_\mathcal{U}(\mathbf{u}|\mathcal{V}^{n+w}_{k_{\text{rand}}})$. Now, consider those iterations $n + w$ with $w \in \mathcal{W}$ such that $k^{n+w}_{\text{rand}} = k^n_{\text{rand}}$ by Lemma 1.1. We will show that the series $\sum_{w\in\mathcal{W}} \mathbb{P}(A^{\text{new},n+w}(\mathbf{u}))$ diverges, and then, we will use the Borel–Cantelli lemma to show that any given $\mathbf{u} \in \mathcal{U}$ will be selected infinitely often to be control input $\mathbf{u}^{n+w}_{\text{new}}$. By Assumption 3.3(i), we have that $\mathbb{P}(A^{\text{new},n+w}(\mathbf{u})) = f^{n+w}_\mathcal{U}(\mathbf{u}|\mathcal{V}^{n+w}_{k_{\text{rand}}})$ is bounded below by a strictly positive constant $\zeta > 0$ for all $w \in \mathcal{W}$. Therefore, we have that $\sum_{w\in\mathcal{W}} \mathbb{P}(A^{\text{new},n+w}(\mathbf{u}))$ diverges, since it is an infinite sum of a strictly positive constant term. Using this result along with the fact that the events $A^{\text{new},n+w}(\mathbf{u})$ are independent, by Assumption 3.3(ii), we get that $\mathbb{P}(\limsup_{w\to\infty} A^{\text{new},n+w}(\mathbf{u})) = 1$, due to the Borel–Cantelli lemma. In words, this means that the events $A^{\text{new},n+w}(\mathbf{u})$ for $w \in \mathcal{W}$ occur infinitely often. Thus, given any subset $\mathcal{V}^n_{k_{\text{rand}}}$, for every control input $\mathbf{u}$ and for all $n \in \mathbb{N}_+$, there exists an infinite subsequence $\mathcal{W}' \subseteq \mathcal{W}$ so that for all $w \in \mathcal{W}'$, it holds $\mathbf{u}^{n+w}_{\text{new}} = \mathbf{u}$. □

Before stating the next result, we first define the *reachable* state space of a state $\mathbf{q}(t) = [\mathbf{p}(t), \mathcal{M}(t), q_D(t)] \in \mathcal{V}^n_k$, denoted by $\mathcal{R}(\mathbf{q}(t))$, that collects all states $\mathbf{q}(t+1) = [\mathbf{p}(t+1), \mathcal{M}(t+1), q_D(t+1)]$ that can be reached within one time step from $\mathbf{q}(t)$.

Corollary A.3 (Reachable set $\mathcal{R}(\mathbf{q})$): Given any state $\mathbf{q} = [\mathbf{p}, \mathcal{M}, q_D] \in \mathcal{V}^n_k$, for any $k \in \{1, \ldots, K_n\}$, all states that belong to the reachable set $\mathcal{R}(\mathbf{q})$ will be added to $\mathcal{V}^{n+w}$, with probability 1, as $w \to \infty$, i.e., $\lim_{w\to\infty} \mathbb{P}(\{\mathcal{R}(\mathbf{q}) \subseteq \mathcal{V}^{n+w}\}) = 1$. Also, edges from $\mathbf{q}(t)$ to all reachable states $\mathbf{q}' \in \mathcal{R}(\mathbf{q})$ will be added to $\mathcal{E}^{n+w}$, with probability 1, as $w \to \infty$, i.e., $\lim_{w\to\infty} \mathbb{P}(\{\cup_{\mathbf{q}'\in\mathcal{R}(\mathbf{q})}(\mathbf{q}, \mathbf{q}') \subseteq \mathcal{E}^{n+w}\}) = 1$.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:45:49 UTC from IEEE Xplore. Restrictions apply.

Proof: We show this result by contradiction. Let $\mathbf{q} \in \mathcal{V}_k^n$ and $\mathbf{q}' \in \mathcal{R}(\mathbf{q})$. Since $\mathbf{q}' \in \mathcal{R}(\mathbf{q})$, we have that there exists a control input $\mathbf{u}$ so that $\mathbf{q}'$ can be reached from $\mathbf{q}$. Assume that the edge from $\mathbf{q}$ to $\mathbf{q}'$ will never be added to the tree. This can happen in the following two cases.

1) There exists no subsequent iteration $n+w$, for some $w > 0$, where the set $\mathcal{V}_k^n$ will be selected to be $\mathcal{V}_{k_{\text{rand}}}^{n+w}$, and therefore, no edges starting from $\mathbf{q}$ will ever be constructed. However, this contradicts Lemma A.1.
2) There exists no subsequent iteration $n+w$, for some $w > 0$, where the control input u will be selected to be $\mathbf{u}_{\text{new}}^{n+w}$, and therefore, the edge from $\mathbf{q}$ to $\mathbf{q}'$ will never be considered to be added to the tree. However, this contradicts Lemma A.2.

Therefore, we conclude that all states that belong to the reachable set $\mathcal{R}(\mathbf{q})$ along with edges from $\mathbf{q}$ to the states in $\mathcal{R}(\mathbf{q})$ will eventually be added to the tree structure. In mathematical terms, this can be written as $\lim_{w\to\infty} \mathbb{P}(\{\mathcal{R}(\mathbf{q}(t)) \subseteq \mathcal{V}^{n+w}\}) = 1$ and $\lim_{w\to\infty} \mathbb{P}(\{\cup_{\mathbf{q}'\in\mathcal{R}(\mathbf{q})}(\mathbf{q},\mathbf{q}') \subseteq \mathcal{E}^{n+w}\}) = 1$, completing the proof.

Corollary A.4: Consider any (feasible or not) path $\mathbf{q}_{0:H} = \mathbf{q}(0), \mathbf{q}(1), \ldots, \mathbf{q}(H)$, where $\mathbf{q}(h) \in \mathcal{R}(\mathbf{q}(h-1))$, for all $h \in \{1, \ldots, H\}$. Then, Algorithm 1 will find $\mathbf{q}_{0:H}$ with probability 1 as $n \to \infty$.

Proof: By construction of the path $\mathbf{q}_{0:H}$, it holds that $\mathbf{q}(h) \in \mathcal{R}(\mathbf{q}(h-1))$, for all $h \in \{1, \ldots, H\}$. Since $\mathbf{q}(0) \in \mathcal{V}^1$, it holds that all states $\mathbf{q} \in \mathcal{R}(\mathbf{q}(0))$, including the state $\mathbf{q}(1)$ (i.e., the second state in the path $\mathbf{q}_{0:H}$), will be added to $\mathcal{V}^n$ with probability 1, as $n \to \infty$, due to Corollary A.3. Once this happens, the edge $(\mathbf{q}(0), \mathbf{q}(1))$ will be added to set of edges $\mathcal{E}^n$ due to Corollary A.3. Applying Corollary A.3 inductively, we get that $\lim_{n\to\infty} \mathbb{P}(\{\mathbf{q}(h) \in \mathcal{V}^n\}) = 1$ and $\lim_{n\to\infty} \mathbb{P}(\{(\mathbf{q}(h-1), \mathbf{q}(h)) \in \mathcal{E}^n\}) = 1$, for all $h \in \{1, \ldots, H\}$, meaning that the path $\mathbf{q}_{0:H}$ will be added to the tree $\mathcal{G}^n$ with probability 1 as $n \to \infty$.

Proof of Theorem 5.1: Probabilistic completeness directly follows from Corollary 1.4. Specifically, since Algorithm 1 can find all possible paths $\mathbf{q}_{0:H}$, it will also find any feasible path (if they exist), which satisfies ϕ.

Proof of Theorem 5.2: The proof of this result follows from Theorem 5.1. Specifically, recall from Theorem 5.1 that Algorithm 1 can find any feasible path and, therefore, the optimal path as well, with probability 1, as $n \to \infty$.

APPENDIX B
DESIGNING BIASED SAMPLING FUNCTIONS

As discussed in Section V, any mass functions $f_{\mathcal{V}}$ and $f_{\mathcal{U}}$ that satisfy Assumptions 3.2 and 3.3, respectively, can be employed in Algorithm 1 without sacrificing its completeness and optimality guarantees. In what follows, we describe a nonuniform/biased sampling strategy, originally developed in [30], allowing us to address large-scale planning tasks.

A. Pruning the DFA

We first prune the DFA by removing transitions that are unlikely to be enabled. Particularly, these are DFA transitions that are enabled when a robot j has to satisfy atomic propositions that require it to be present in more than one location *simultaneously*. Hereafter, these DFA transitions are called *infeasible*. For instance, a DFA transition that is enabled if a robot is required to be simultaneously in more than one known disjoint region of interest is classified as infeasible; see [15] for more details. Determining infeasible transitions becomes more challenging for transitions that depend on probabilistic predicates as, e.g., in (6). The reason is that searching over the space of distributions is required to determine if there exist robot positions that satisfy a Boolean formula defined over such predicates; recall that the map distribution is updated during the execution of Algorithm 1. Hereafter, for simplicity, we classify such DFA transitions that require a robot to be close to more than one landmark at the same time as *infeasible*.[9]

Second, along the lines of [15], we define a distance function $d : \mathcal{Q}_D \times \mathcal{Q}_D \to \mathbb{N}$ between any two DFA states, which captures the minimum number of transitions in the pruned DFA to reach a state q'_D from a state q_D. This function is defined as follows:

$$d(q_D, q'_D) = \begin{cases} |SP_{q_D,q'_D}|, & \text{if } SP_{q_D,q'_D} \text{ exists} \\ \infty, \text{otherwise} \end{cases} \tag{17}$$

where SP_{q_D,q'_D} denotes the shortest path (in terms of hops) in the pruned DFA from q_D to q'_D and $|SP_{q_D,q'_D}|$ stands for its cost (number of hops).

B. Mass Function $f_{\mathcal{V}}$

To construct a nonuniform mass function $f_{\mathcal{V}}$, first, we define the set $\mathcal{D}_{\text{min}}$ that collects all tree nodes $\mathbf{q} = (\mathbf{p}, \mathcal{M}, q_D)$ with the smallest distance $d(q_D, q_F)$, denoted by d_{min}, among all nodes in the tree, i.e., $\mathcal{D}_{\text{min}} = \{\mathbf{q} = (\mathbf{p}, \mathcal{M}, q_D) \in \mathcal{V} | d(q_D, q_F) = d_{\text{min}}\}$. The set $\mathcal{D}_{\text{min}}$ initially collects only the root and is updated (along with d_{min}) as new states are added to the tree. Given the set $\mathcal{D}_{\text{min}}$, we define the set $\mathcal{K}_{\text{min}}$ that collects the indices k of the subsets $\mathcal{V}_k$ that satisfy $\mathcal{V}_k \cap \mathcal{D}_{\text{min}} \neq \emptyset$. Given the set $\mathcal{K}_{\text{min}}$, the probability mass function $f_{\text{rand}}(k|\mathcal{V})$ is defined so that it is biased to select more often subsets $\mathcal{V}_k \subseteq \mathcal{V}$ that satisfy $k \in \mathcal{K}_{\text{min}}$. Specifically, $f_{\text{rand}}(k|\mathcal{V})$ is defined as follows:

$$f_{\text{rand}}(k|\mathcal{V}) = \begin{cases} p_{\text{rand}} \frac{1}{|\mathcal{K}_{\text{min}}|}, \text{if} k \in \mathcal{K}_{\text{min}} \\ (1 - p_{\text{rand}}) \frac{1}{|\mathcal{V} \setminus \mathcal{K}_{\text{min}}|}, \text{otherwise} \end{cases} \tag{18}$$

where $p_{\text{rand}} \in (0.5, 1)$ stands for the probability of selecting *any* subset $\mathcal{V}_k$ that satisfies $k \in \mathcal{K}_{\text{min}}$. Note that p_{rand} can change with iterations n, but it should always satisfy $p_{\text{rand}} \in (0.5, 1)$ to ensure that subsets $\mathcal{V}_k$ with $k \in \mathcal{K}_{\text{min}}$ are selected more often.

C. Mass Function $f_{\mathcal{U}}$

The sampling function $f_{\mathcal{U}}(\mathbf{u}|k_{\text{rand}})$ is designed so that a state $\mathbf{q} = (\mathbf{p}, \mathcal{M}, q_F)$ is reached by following the shortest path (in

[9]Note that this is a conservative approach as transitions that are in fact feasible may be classified as infeasible transitions. Nevertheless, this does not affect the correctness of the proposed algorithm, since the biased sampling functions that will be constructed next still satisfy Assumptions 3.2 and 3.3. If in the pruned DFA, there is no path from the initial to the final state, then this means that either the specification is infeasible or the task is feasible, but the pruning was very conservative. In this case, a warning message can be returned to the user, and either uniform distributions or the proposed biased distributions can be employed but using the original (and not the pruned) DFA.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:45:49 UTC from IEEE Xplore. Restrictions apply.

terms of hops) in the pruned DFA that connects q_D^0 to q_F. First, given a state $\mathbf{q}_{\text{rand}} = (\mathbf{p}_{\text{rand}}, \mathcal{M}_{\text{rand}}, q_D^{\text{rand}})$, we compute the next DFA state defined as $q_D^{\text{next}} = \delta_D(q_D^{\text{rand}}, L(\mathbf{p}_{\text{rand}}, \mathcal{M}_{\text{rand}}))$. Next, we construct the reachable set $\mathcal{R}_D(q_D^{\text{next}})$ that collects all states $q_D \in \mathcal{Q}_D$ that can be reached in one hop in the *pruned* DFA from q_D^{next}, for any observation/symbol $\sigma \in 2^{\mathcal{AP}}$, defined as $\mathcal{R}_D(q_D^{\text{next}}) = \{q_D \in \mathcal{Q}_D | \exists \sigma \in 2^{\mathcal{AP}} \text{s.t.} \delta_D(q_D^{\text{next}}, \sigma) = q_D\}$. Among all states in $\mathcal{R}_D(q_D^{\text{next}})$, we select the state, denoted by q_D^{min}, with the minimum distance from q_F.

Given q_D^{next} and q_D^{min}, we select a symbol σ that enables a transition from q_D^{next} to q_D^{min} in the pruned DFA. Based on the selected symbol σ, we select locations that if every robot visits, then σ is generated and transition from q_D^{next} to q_D^{min} is achieved. To this end, first, we select the atomic proposition that appears in σ and is associated with robot j. Note that by definition of the feasible transitions and by construction of the pruned DFA, there is only one (if any) atomic proposition in σ related to robot j. With slight abuse of notation, we denote this atomic proposition by $\pi_j \in \mathcal{AP}$. For instance, the atomic proposition π_j corresponding to the word $\sigma = \pi_p(\mathbf{p}, \mathcal{M}, \{j, r_1, \delta_1, c_1\})\pi_p(\mathbf{p}, \mathcal{M}, \{z, r_2, \delta_2, c_2\})$ is $\pi_j = \pi_p(\mathbf{p}, \mathcal{M}, \{j, r_1, \delta_1, c_1\})$. In addition, observe that there is no atomic proposition π_s associated with any robot $s \neq j, z$. Next, given π_j, we select a landmark that if robot j approaches/localizes, then π_j may be satisfied. The selected landmark is denoted by L_j and is used to design $f_{\text{new},i}(\mathbf{u}_i|k)$ so that control inputs that drive robot j toward L_j are selected more often than others. Note that for robots j for which there is no predicate π_j associated with them in σ, we have that $L_j = \varnothing$. In other words, the location of such robots j does not play any role in generating the word σ. For instance, following the previous example, L_j is selected to be the landmark that has the highest probability of having the label c_1, based on the discrete distribution $\mathbf{d}$, while for all other robots $s \neq j, z$, it holds $L_s = \varnothing$.

Given a landmark L_j, we construct the mass function $f_{\text{new},j}(\mathbf{u}_j|k)$, from which we sample a control input $\mathbf{u}_j$ as follows:

$$f_{\text{new},j}(\mathbf{u}_j|k) = \begin{cases} \frac{1}{|\mathcal{U}_j|}, \text{if} L_j = \varnothing \\ p_{\text{new}}, \text{if}(L_j \neq \varnothing) \wedge (\mathbf{u}_j = \mathbf{u}_j^*) \wedge (d_j \geq R_j) \\ (1 - p_{\text{new}}) \frac{1}{|\mathcal{U}_j \setminus \{\mathbf{u}_j^*\}|}, \text{if}(L_j \neq \varnothing) \\ \wedge (\mathbf{u}_j \neq \mathbf{u}_j^*) \wedge (d_j \leq R_j) \end{cases} \tag{19}$$

where: 1) d_j denotes the *geodesic* distance between robot j and the estimated position of landmark L_j, i.e., $d_j = \|\hat{\mathbf{x}}_{L_j}(t+1) - \mathbf{p}_j(t+1)\|_g$ [74]; 2) R_j denotes the sensing range of robot j; and 3) $\mathbf{u}_j^* \in \mathcal{U}_j$ is the control input that minimizes the geodesic distance between $\mathbf{p}_j(t+1)$ and $\hat{\mathbf{x}}_i(t+1)$, i.e., $\mathbf{u}_j^* = \text{argmin}_{\mathbf{u}_j \in \mathcal{U}_j} \|\hat{\mathbf{x}}_i(t+1) - \mathbf{p}_j(t+1)\|_g$, where $\|\cdot\|_g$ denotes the geodesic norm/distance. In words, in the first case of (19), a control input $\mathbf{u}_j$ is drawn from a uniform distribution if $L_j = \varnothing$. In the second case, the input $\mathbf{u}_j^* \in \mathcal{U}_j$ is selected, if the (predicted) distance between robot j and the assigned landmark L_j is greater than the sensing range. In the third case, if the assigned landmark is within the sensing range, random inputs are selected to get different views of the assigned landmark, which will contribute to decreasing the uncertainty of the assigned landmark.

Finally, to compute the geodesic distance between $\mathbf{p}_j(t+1)$ and the estimated location of L_j, we first treat as "virtual/temporal" obstacles all landmarks that if visited, the Boolean condition under which transition from q_D^{next} to q_D^{min} is enabled is false. For example, assume that transition from q_D^{next} to q_D^{min} is possible if the following Boolean condition is true $\pi_p(\mathbf{p}, \mathcal{M}, \{j, r_1, \delta_1, c_1\}) \wedge \pi_p(\mathbf{p}, \mathcal{M}, \{z, r_2, \delta_2, c_2\}) \wedge [\neg \pi_p(\mathbf{p}, \mathcal{M}, \{j, \ell_1, r_1, \delta_1\})]$, which is satisfied by the following symbol $\sigma = \pi_p(\mathbf{p}, \mathcal{M}, \{j, r_1, \delta_1, c_1\})\pi_p(\mathbf{p}, \mathcal{M}, \{z, r_2, \delta_2, c_2\})$ and is violated if $\pi_p(\mathbf{p}, \mathcal{M}, \{j, \ell_1, r_1, \delta_1\}$ is observed, i.e., if robot j approaches the landmark ℓ_1 as per the probabilistic requirements captured by $\pi_p(\mathbf{p}, \mathcal{M}, \{j, \ell_1, r_1, \delta_1\}$. For such "virtual" obstacles, we first compute a ϵ-confidence interval around the mean position of landmark ℓ_1, for some $\epsilon > 0$ (e.g., $\epsilon = 90\%$). This ϵ-confidence interval is then treated as a (known) physical obstacle in the workspace. Then, we compute the *geodesic* distance between $\mathbf{p}_j(t+1)$ and such virtual obstacles. Essentially, this empirical approach minimizes the probability of generating an observation that violates the assigned task specification. Observe that the "virtual" obstacles are different for each robot, while they may also depend on the global state $\mathbf{p}(t)$ (and not solely on individual robot states $\mathbf{p}_j(t)$). For instance, consider the Boolean condition $\pi_p(\mathbf{p}, \mathcal{M}, \{j, r_1, \delta_1, c_1\}) \wedge \pi_p(\mathbf{p}, \mathcal{M}, \{z, r_2, \delta_2, c_2\}) \wedge \{[\neg \pi_p(\mathbf{p}, \mathcal{M}, \{j, \ell_1, r_1, \delta_1\})] \wedge [\neg \pi_p(\mathbf{p}, \mathcal{M}, \{z, \ell_2, r_2, \delta_2\})]\}$ This is violated only if *both* robots j and z are sufficiently close to landmarks ℓ_1 and ℓ_2, respectively. Such "virtual" obstacles are ignored, i.e., in this case, ℓ_1 is not treated as an obstacle for robot j.

REFERENCES

[1] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[2] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for mobile robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, Barcelona, Spain, Apr. 2005, pp. 2020–2025.

[3] M. Guo and M. M. Zavlanos, "Distributed data gathering with buffer constraints and intermittent communication," in *Proc. IEEE Int. Conf. Robot. Automat.*, Singapore, 2017, pp. 279–284.

[4] K. Leahy, D. Zhou, C.-I. Vasile, K. Oikonomopoulos, M. Schwager, and C. Belta, "Persistent surveillance for unmanned aerial vehicles subject to charging and temporal logic constraints," *Auton. Robots*, vol. 40, no. 8, pp. 1363–1378, 2016.

[5] C. Baier and J.-P. Katoen, *Principles of Model Checking*. Cambridge, MA, USA: MIT Press, 2008.

[6] E. M. Clarke, O. Grumberg, and D. Peled, *Model Checking*. Cambridge, MA, USA: MIT Press, 1999.

[7] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, "Hybrid controllers for path planning: A temporal logic approach," in *Proc. 44th IEEE Conf. Decis. Control, Eur. Control Conf.*, Seville, Spain, 2005, pp. 4885–4890.

[8] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Where's Waldo? Sensor-based temporal logic motion planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2007, pp. 3116–3121.

[9] S. L. Smith, J. Tumova, C. Belta, and D. Rus, "Optimal path planning for surveillance with temporal-logic constraints," *Int. J. Robot. Res.*, vol. 30, no. 14, pp. 1695–1708, 2011.

[10] J. Tumova and D. V. Dimarogonas, "Multi-agent planning under local LTL specifications and event-based synchronization," *Automatica*, vol. 70, pp. 239–248, 2016.

[11] Y. Chen, X. C. Ding, A. Stefanescu, and C. Belta, "Formal approach to the deployment of distributed robotic teams," *IEEE Trans. Robot.*, vol. 28, no. 1, pp. 158–171, Feb. 2012.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:45:49 UTC from IEEE Xplore. Restrictions apply.

[12] A. Ulusoy, S. L. Smith, and C. Belta, "Optimal multi-robot path planning with LTL constraints: Guaranteeing correctness through synchronization," in *Distributed Autonomous Robotic Systems*. New York, NY, USA: Springer, 2014, pp. 337–351.

[13] Y. Shoukry *et al.*, "Linear temporal logic motion planning for teams of underactuated robots using satisfiability modulo convex programming," in *Proc. IEEE 56th Annu. Conf. Decis. Control*, Dec. 2017, pp. 1132–1137.

[14] C. I. Vasile and C. Belta, "Sampling-based temporal logic path planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Tokyo, Japan, Nov. 2013, pp. 4817–4822.

[15] Y. Kantaros and M. M. Zavlanos, "STyLuS*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems," *Int. J. Robot. Res.*, vol. 39, pp. 812–836, 2020.

[16] D. Sadigh, E. S. Kim, S. Coogan, S. S. Sastry, and S. A. Seshia, "A learning based approach to control synthesis of Markov decision processes for linear temporal logic specifications," in *Proc. 53rd IEEE Conf. Decis. Control*, 2014, pp. 1091–1096.

[17] X. Li, C.-I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 3834–3839.

[18] A. Jones, D. Aksaray, Z. Kong, M. Schwager, and C. Belta, "Robust satisfaction of temporal logic specifications via reinforcement learning," 2015, *arXiv:1510.06460*.

[19] M. Hasanbeig, Y. Kantaros, A. Abate, D. Kroening, G. J. Pappas, and I. Lee, "Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees," in *Proc. 58th IEEE Conf. Decis. Control*, Nice, France, Dec. 2019, pp. 5338–5343.

[20] A. Jones, M. Schwager, and C. Belta, "Distribution temporal logic: Combining correctness with quality of estimation," in *Proc. 52nd IEEE Conf. Decis. Control*, 2013, pp. 4719–4724.

[21] S. Haesaert *et al.*, "Temporal logic control of POMDPs via label-based stochastic simulation relations," *IFAC-PapersOnLine*, vol. 51, no. 16, pp. 271–276, 2018.

[22] A. Bianco and L. De Alfaro, "Model checking of probabilistic and nondeterministic systems," in *Foundations of Software Technology and Theoretical Computer Science*. New York, NY, USA: Springer, 1995, pp. 499–513.

[23] R. Donaldson and D. Gilbert, "A monte carlo model checker for probabilistic LTL with numerical constraints," Dept. Comput. Sci., Univ. Glasgow, Glasgow, U.K., Res. Rep. TR-2008-282, 2008.

[24] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theor. Comput. Sci.*, vol. 410, no. 42, pp. 4262–4291, 2009.

[25] C. Yoo and C. Belta, "Control with probabilistic signal temporal logic," 2015, *arXiv:1510.08474*.

[26] L. Lindemann and D. V. Dimarogonas, "Barrier function based collaborative control of multiple robots under signal temporal logic tasks," *IEEE Control Netw. Syst.*, vol. 7, no. 4, pp. 1916–1928, Dec. 2020.

[27] K. Sun, B. Schlotfeldt, G. J. Pappas, and V. Kumar, "Stochastic motion planning under partial observability for mobile robots with continuous range measurements," *IEEE Trans. Robot.*, vol. 37, no. 3, pp. 979–995, Jun. 2021.

[28] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Proc. IEEE Int. Conf. Robot. Automat.*, 1997, vol. 3, pp. 2719–2726.

[29] I. A. Sucan and L. E. Kavraki, "A sampling-based tree planner for systems with complex dynamics," *IEEE Trans. Robot.*, vol. 28, no. 1, pp. 116–131, Feb. 2012.

[30] Y. Kantaros and G. J. Pappas, "Optimal temporal logic planning for multi-robot systems in uncertain semantic maps," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Macao, Nov. 2019, pp. 4127–4132.

[31] N. Atanasov, J. L. Ny, K. Daniilidis, and G. J. Pappas, "Information acquisition with sensing robots: Algorithms and error bounds," in *Proc. IEEE Int. Conf. Robot. Automat.*, Hong Kong, 2014, pp. 6447–6454.

[32] N. Atanasov, J. L. Ny, K. Daniilidis, and G. J. Pappas, "Decentralized active information acquisition: Theory and application to multi-robot SLAM," in *Proc. IEEE Int. Conf. Robot. Automat.*, Seattle, WA, USA, 2015, pp. 4775–4782.

[33] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas, "Probabilistic data association for semantic SLAM," in *Proc. IEEE Int. Conf. Robot. Automat.*, Singapore, May/Jun. 2017, pp. 1722–1729.

[34] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: An open-source library for real-time metric-semantic localization and mapping," in *Proc. IEEE Int. Conf. Robot. Automat.*, Paris, France, Jun. 2020, pp. 1689–1696.

[35] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning with deterministic μ-calculus specifications," in *Proc. Amer. Control Conf.*, Montreal, QC, Canada, Jun. 2012, pp. 735–742.

[36] A. Bhatia, L. E. Kavraki, and M. Y. Vardi, "Sampling-based motion planning with temporal goals," in *Proc. Int. Conf. Robot. Automat.*, Anchorage, AK, USA, 2010, pp. 2689–2696.

[37] 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.

[38] X. Luo, Y. Kantaros, and M. M. Zavlanos, "An abstraction-free method for multirobot temporal logic optimal control synthesis," *IEEE Trans. Robot.*, vol. 37, no. 5, pp. 1487–1507, Oct. 2021.

[39] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[40] C. Belta, V. Isler, and G. J. Pappas, "Discrete abstractions for robot motion planning and control in polygonal environments," *IEEE Trans. Robot.*, vol. 21, no. 5, pp. 864–874, Oct. 2005.

[41] M. Guo, K. H. Johansson, and D. V. Dimarogonas, "Revising motion planning under linear temporal logic specifications in partially known workspaces," in *Proc. IEEE Int. Conf. Robot. Automat.*, Karlsruhe, Germany, 2013, pp. 5025–5032.

[42] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local LTL specifications," *Int. J. Robot. Res.*, vol. 34, no. 2, pp. 218–235, 2015.

[43] M. R. Maly, M. Lahijanian, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi, "Iterative temporal motion planning for hybrid systems in partially unknown environments," in *Proc. 16th Int. Conf. Hybrid Syst., Comput. Control*, 2013, pp. 353–362.

[44] M. Lahijanian, M. R. Maly, D. Fried, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi, "Iterative temporal planning in uncertain environments with partial satisfaction guarantees," *IEEE Trans. Robot.*, vol. 32, no. 3, pp. 583–599, Jun. 2016.

[45] S. C. Livingston, R. M. Murray, and J. W. Burdick, "Backtracking temporal logic synthesis for uncertain environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2012, pp. 5163–5170.

[46] S. C. Livingston, P. Prabhakar, A. B. Jose, and R. M. Murray, "Patching task-level robot controllers based on a local μ-calculus formula," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2013, pp. 4588–4595.

[47] Y. Kantaros, M. Malencia, V. Kumar, and G. J. Pappas, "Reactive temporal logic planning for multiple robots in unknown environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, Paris, France, Jun. 2020, pp. 11479–11485.

[48] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao-Blackwellized particle filters," *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 34–46, Feb. 2007.

[49] G. Pola, A. Girard, and P. Tabuada, "Approximately bisimilar symbolic models for nonlinear control systems," *Automatica*, vol. 44, no. 10, pp. 2508–2516, 2008.

[50] A. Desai, S. Ghosh, S. A. Seshia, N. Shankar, and A. Tiwari, "SOTER: A runtime assurance framework for programming safe robotics systems," in *Proc. 49th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, 2019, pp. 138–150.

[51] J. Fu, N. Atanasov, U. Topcu, and G. J. Pappas, "Optimal temporal logic planning in probabilistic semantic maps," in *Proc. IEEE Int. Conf. Robot. Automat.*, Stockholm, Sweden, 2016, pp. 3690–3697.

[52] C. Freundlich, S. Lee, and M. M. Zavlanos, "Distributed active state estimation with user-specified accuracy," *IEEE Trans. Automat. Control*, vol. 63, no. 2, pp. 418–433, Feb. 2018.

[53] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.

[54] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[55] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1321–1330.

[56] N. Atanasov, M. Zhu, K. Daniilidis, and G. J. Pappas, "Localization from semantic observations via the matrix permanent," *Int. J. Robot. Res.*, vol. 35, nos. 1–3, pp. 73–99, 2016.

[57] B. Luders, M. Kothari, and J. How, "Chance constrained RRT for probabilistic robustness to environmental uncertainty," in *Proc. AIAA Guid., Navigat., Control Conf.*, 2010, Art. no. 8160.

[58] M. Lahijanian and M. Kwiatkowska, "Specification revision for Markov decision processes with optimal trade-off," in *Proc. IEEE 55th Conf. Decis. Control*, 2016, pp. 7411–7418.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:45:49 UTC from IEEE Xplore. Restrictions apply.

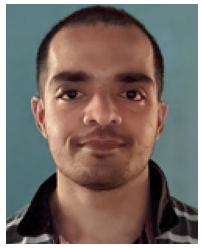
[59] M. Narasimhan *et al.*, "Seeing the un-scene: Learning amodal semantic maps for room navigation," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 513–529.
[60] E. Yel and N. Bezzo, "GP-based runtime planning, learning, and recovery for safe UAV operations under unforeseen disturbances," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 2173–2180.
[61] L. Lindemann, M. Cleaveland, Y. Kantaros, and G. J. Pappas, "Robust motion planning in the presence of estimation uncertainty," in *Proc. IEEE Conf. Decis. Control*, Austin, TX, USA, Dec. 2021.
[62] M. Ryll, J. Ware, J. Carter, and N. Roy, "Efficient trajectory planning for high speed flight in unknown environments," in *Proc. Int. Conf. Robot. Automat.*, Montreal, QC, Canada, May 2019, pp. 732–738.
[63] A. Elhafsi, B. Ivanovic, L. Janson, and M. Pavone, "Map-predictive motion planning in unknown environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 8552–8558.
[64] K. Y. Leung, T. D. Barfoot, and H. H. Liu, "Decentralized cooperative SLAM for sparsely-communicating robot networks: A centralized-equivalent approach," *J. Intell. Robot. Syst.*, vol. 66, no. 3, pp. 321–342, 2012.
[65] Y. Kantaros, B. Schlotfeldt, N. Atanasov, and G. J. Pappas, "Asymptotically optimal planning for non-myopic multi-robot information gathering," in *Proc. Robot., Sci. Syst.*, Jun. 2019, pp. 22–26.
[66] T. Summers, "Distributionally robust sampling-based motion planning under uncertainty," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 6518–6523.
[67] V. Renganathan, I. Shames, and T. H. Summers, "Towards integrated perception and motion planning with distributionally robust risk constraints," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 15530–15536, 2020.
[68] "Simulation videos: Safe planning over uncertain semantic maps," Accessed: Oct. 14, 2021. [Online]. Available: https://youtu.be/mVW_7ZLpQgA
[69] B. Schlotfeldt, D. Thakur, N. Atanasov, V. Kumar, and G. J. Pappas, "Anytime planning for decentralized multirobot active information gathering," *IEEE Robot. Automat. Lett.*, vol. 3, no. 2, pp. 1025–1032, Apr. 2018.
[70] A. Duret-Lutz and D. Poitrenaud, "SPOT: An extensible model checking library using transition-based generalized Bu/spl uml/chi automata," in *Proc. IEEE Comput. Soc. 12th Annu. Int. Symp. Model., Anal., Simul. Comput. Telecommun. Syst.*, 2004, pp. 76–83.
[71] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "RotorS—A modular gazebo MAV simulator framework," in *Robot Operating System (ROS): The Complete Reference*, vol. 1. Cham, Switzerland: Springer, 2016, pp. 595–625.
[72] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 2520–2525.
[73] G. Grimmett and D. Stirzaker, *Probability and Random Processes*. Oxford, U.K.: Oxford Univ. Press, 2001.
[74] Y. Kantaros and M. M. Zavlanos, "Global planning for multi-robot communication networks in complex environments," *IEEE Trans. Robot.*, vol. 32, no. 5, pp. 1045–1061, Oct. 2016.



Yiannis Kantaros (Member, IEEE) received the Diploma degree in electrical and computer engineering from the University of Patras, Patras, Greece, in 2012, and the M.Sc. and Ph.D. degrees in mechanical engineering from Duke University, Durham, NC, USA, in 2017 and 2018, respectively.

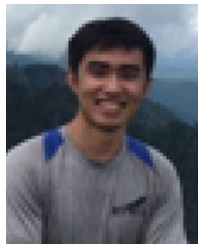
He is currently an Assistant Professor with the Department of Electrical and Systems Engineering, Washington University in St. Louis (WashU), St. Louis, MO, USA. Prior to joining WashU, he was a Postdoctoral Associate with the Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, USA. His current research interests include machine learning, distributed control and optimization, and formal methods with applications in robotics.

Dr. Kantaros is the recipient of the Best Student Paper Award at the 2nd IEEE Global Conference on Signal and Information Processing in 2014 and the 2017–2018 Outstanding Dissertation Research Award from the Department of Mechanical Engineering and Materials Science, Duke University.



Samarth Kalluraya received the B.E. degree in mechanical engineering from the University of Pune, Pune, India, in 2018, and the M.S.E. degree in mechanical engineering and applied mechanics from the University of Pennsylvania, Philadelphia, PA, USA, in 2020. He is currently working toward the Ph.D. degree in electrical and systems engineering with Washington University in St. Louis, St. Louis, MO, USA.

His research interests include motion planning, environment mapping, and machine learning.



Qi Jin received the B.E. degree in mechanical engineering from the University of Illinois at Urbana Champaign, Champaign, IL, USA, in 2018, and the M.S.E. degree in robotics from the University of Pennsylvania, Philadelphia, PA, USA, in 2021.



George J. Pappas (Fellow, IEEE) received the Ph.D. degree in electrical engineering and computer sciences from the University of California, Berkeley, CA, USA, in 1998.

He is currently the Joseph Moore Professor and the Chair with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, USA, where he also holds a secondary appointment with the Department of Computer and Information Sciences and the Department of Mechanical Engineering and Applied Mechanics and is a member of the General Robotics, Automation, Sensing and Perception Lab and the PRECISE Center. He had previously served as the Deputy Dean for Research with the School of Engineering and Applied Science, University of Pennsylvania. His research interests include control theory and, in particular, hybrid systems, embedded systems, cyber-physical systems, and hierarchical and distributed control systems, with applications to unmanned aerial vehicles, distributed robotics, green buildings, and biomolecular networks.

Dr. Pappas is the recipient of various awards, such as the Antonio Ruberti Young Researcher Prize, the George S. Axelby Award, the Hugo Schuck Best Paper Award, the George H. Heilmeier Award, the National Science Foundation Presidential Early Career Award for Scientists and Engineers, and numerous Best Student Papers Awards.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:45:49 UTC from IEEE Xplore. Restrictions apply.