Journal Title
XX(X):1–20
©The Author(s) 0000
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Control in Belief Space with Temporal Logic Specifications using Vision-based Localization

Kevin Leahy[1*], Eric Cristofalo[2*], Cristian-Ioan Vasile[3*], Austin Jones[1], Eduardo Montijano[4], Mac Schwager[2], and Calin Belta[5]

Abstract

We present a solution for operating a vehicle without global positioning infrastructure while satisfying constraints on its temporal behavior, and on the uncertainty of its position estimate. The proposed solution is an end-to-end framework for mapping an unknown environment using aerial vehicles, synthesizing a control policy for a ground vehicle in that environment, and using a quadrotor to localize the ground vehicle within the map while it executes its control policy. This vision-based localization is noisy, necessitating planning in the belief space of the ground robot. The ground robot's mission is given using a language called *Gaussian Distribution Temporal Logic (GDTL)*, an extension of Boolean logic that incorporates temporal evolution and noise mitigation directly into the task specifications. We use a sampling-based algorithm to generate a transition system in the belief space and use local feedback controllers to break the *curse of history* associated with belief space planning. To localize the vehicle, we build a high-resolution map of the environment by flying a team of aerial vehicles in formation with sensor information provided by their onboard cameras. The control policy for the ground robot is synthesized under temporal and uncertainty constraints given the semantically labeled map. Then the ground robot can execute the control policy given pose estimates from a dedicated aerial robot that tracks and localizes the ground robot. The proposed method is validated using two quadrotors to build a map, followed by a two-wheeled ground robot and a quadrotor with a camera for ten successful experimental trials.

Keywords

Persistent Specifications, Distribution Temporal Logic, Belief Space Planning, Homography-based Formation Control, Vision-based Localization, Sampling-based Planning

1 Introduction

In this paper, we propose a solution to the following problem: localize and control a noisy ground robot under temporal logic (TL) specifications in an a priori unknown environment with no global positioning infrastructure. Robots operating in the real world typically require accurate pose estimates to compute effective control actions, but in many cases, such as dense urban environments (Hsieh et al. 2007), indoors, or on other planets, GPS may be unavailable or unreliable. Furthermore, it is advantageous to consider an aerial robot for on-the-fly tracking of the ground robot because it can aid in terms of localization as well as obstacle avoidance, leaving the ground robot dedicated to other tasks.

Consider a robot that must perform the following task in an outdoor disaster site: "Periodically collect soil samples from the forest, then the beach. Deliver samples to researchers. Go to a charging station after tasks are complete. Always avoid known obstacles and restricted zones. Ensure that the uncertainty (measured by variance) of the robot's pose is always below 1 m^2." Such a task may be specified using Gaussian distribution TL (GDTL) (Vasile et al. 2016), a specification language that incorporates the robot's desired position as well as uncertainty. Unfortunately, the initial position of the robot is completely unknown and common methods to synthesize

[*]These authors contributed equally to this work.
[1]MIT Lincoln Laboratory
[2]Stanford University
[3]Massachusetts Institute of Technology
[4]Universidad de Zaragoza
[5]Boston University

Corresponding author:
Kevin Leahy, 110 Cummington Mall, Boston MA 02215
Email: kjleahy@bu.edu

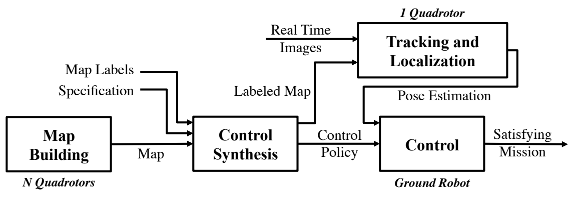


Figure 1. The proposed framework includes three major components: 1) mapping in unknown environments, 2) control synthesis, and 3) online tracking and localization of a ground robot.

a control policy for the robot, even while operating under observation noise, will not be sufficient.

We propose an end-to-end framework (see Fig. 1) that includes a two-wheeled ground robot and a team of aerial robots, i.e., N quadrotors, each equipped with a downward facing camera, an IMU, and an altimeter. The team of quadrotors are first responsible for building the map of the unknown environment using their onboard camera images. Then a closed-loop control policy is computed, given the map, for the ground robot to satisfy its mission specification. Finally, the ground robot operates under the computed control policy with the measurements provided by a single quadrotor tracking it from above. Specifically, these three phases can be described as follows:

1. Generate a mosaic map image of the unknown environment using purely vision and homography-based formation control (Montijano et al. 2016) with multiple quadrotors.
2. Label the generated map and define the mission specification (to be completed by human operator) and then automatically synthesize a satisfying control policy for ground robot using GDTL-Feedback Information RoadMaps, or GDTL-FIRM (Vasile et al. 2016).
3. Simultaneously track and localize the ground robot with a single aerial vehicle using a homography-based pose estimation and position-based visual servoing control method.

This work considers the cooperation between ground and air vehicles and leverages their heterogeneous capabilities to jointly carry out a mission. While other research exists for cooperation among mixed teams of ground and air vehicles, existing research assumes the presence of GPS on either the ground vehicles (Vaughan et al. 2000) or on the aerial vehicles (Hsieh et al. 2007; Grocholsky et al. 2006). We, on the other hand, assume the robots are working in an environment with no external positioning framework whatsoever. Other work that has focused on planning without GPS, such as Forster et al. (2013), uses the visual capabilities of an aerial vehicle to enhance a map built by a ground vehicle. In Mueggler et al. (2014), an aerial and ground vehicle work together without GPS, but do not satisfy a complex mission such as those that can be specified using GDTL. In this work, we assume the map is built by a team of aerial vehicles using their high vantage point so that the ground vehicle can perform a specific task based on the resulting map. Further, unlike these works, in our work, the mission to be carried out is specified using GDTL, allowing for the encoding of much more complex missions, including specifying constraints about the uncertainty of the ground vehicle's localization.

In our solution, we utilize a vision-based mapping process to first map the unknown environment prior to executing the mission. The GDTL specification quantifying the robot's success in the mission is then defined using this map. Vision-based mapping via aerial cameras allows for accurate pose estimation in complicated environments while only employing cheap, readily-available RGB cameras. For example, homography-based visual servoing methods provide accurate localization with only the use of camera data (Benhimane and Malis 2006). In this work, we make use of homography-based consensus control methods (Montijano et al. 2016) for the aerial vehicles to build a mosaic map, and monitor the ground robot with a Position-Based Visual Servoing (PBVS) control method designed to keep the robot in the field of view at all times while guaranteeing sufficient overlap with the map.

We exploit sampling-based techniques to synthesize switched closed-loop control policies that are guaranteed to drive a robot with observation noise while achieving high-level tasks given as temporal logic formulae. Observation and actuation noise are inherent to ground robot motion, hence we provide a temporal logic formulation that directly accounts for this uncertainty. We use Gaussian Distribution Temporal Logic, which is built from traditional Temporal Logics but augmented to include state uncertainty. Traditionally, temporal logic formulae interleave Boolean logic and temporal operators with system properties to specify rich global behaviors. In the domain of robotics, an example of a task that can be encoded in temporal logic is "Periodically clean the living room and then the bathroom. Put the trash in the bin in the kitchen or outside. Go to a charging station after cleaning is complete. Always avoid the bedroom." In the absence of observation noise, tools from formal synthesis can be used to synthesize control policies that ensure these rich specifications are met (Maly et al. 2013; Lahijanian et al. 2016). Temporal logic specifications are much richer than the constraints typically found in planning problems, e.g., problems in which a path is planned to reach a goal while avoiding obstacles (Kaelbling et al. 1998; van den Berg et al. 2011; Hauser 2011; Bachrach et al. 2012; Vitus and Tomlin 2011; Lesser and Oishi 2015). In this work, we present an automatic, hierarchical control synthesis algorithm that extends tools from formal synthesis and stochastic control to enforce temporal logic specifications. We evaluate our algorithm with experiments using a wheeled robot with noisy actuators localized by an aerial robot with a camera while performing a persistent navigation task.

While synthesizing control policies to enforce temporal logic properties under dynamics noise has been extensively considered in the literature (Zamani et al. 2014), observation noise has only recently been considered (Maly et al. 2013; Jones et al. 2013; Leahy et al. 2015; Svorenova et al. 2013; Ayala et al. 2014). One of the technical challenges of incorporating observation noise into formal synthesis is that satisfaction of temporal logic properties is in general defined with respect to the state trajectory of the system rather than the evolution of the belief (as measured by a posterior probability distribution) about this state. In this paper, we introduce the paradigm of Gaussian distribution temporal logic (GDTL) which allows us to specify properties involving the uncertainty in the state of the system, e.g. "Ensure that the uncertainty (measured by variance) of the robot's x position is always below 0.1 m^2". GDTL formulae can be translated to Rabin automata using off-the-shelf tools (Jones et al. 2013).

The problem of synthesizing controllers to enforce a GDTL specification is in general a discrete time, continuous space partially observable Markov decision process (POMDP). Our approach approximates the optimal solution with a computationally feasible hierarchical sampling-based control synthesis algorithm. Most existing sampling-based algorithms sample points directly in belief space (Patil et al. 2015; Burns and Brock 2007; Bry and Roy 2011; Prentice and Roy 2009), which requires synthesizing distribution-to-distribution controllers. Such synthesis problems are computationally difficult and may require significant modeling on the part of a control designer. To circumvent these challenges, we base the core of our algorithm on feedback information roadmaps (FIRMs). The FIRM motion planner extends probabilistic roadmaps (PRMs) (Thrun et al. 2005), to handle observation noise. In FIRM, points are sampled directly in the state space (rather than in belief space) and feedback control policies, e.g. linear quadratic Gaussian (LQG) controllers, stabilize the system about nodes along paths in the roadmap. The behavior of the closed-loop system is then used to predict how the state estimate evolves. The associated trajectories of the estimate induce a roadmap in the belief space.

If the goal of the problem were only to reach a given region of the belief space, one could construct a switched controller by finding a path in the roadmap from the initial distribution to a node contained within the goal set and then applying the corresponding sequence of controllers. During the application of the controller, however, we do not have any guarantees about whether or not the evolution of the system will violate the given specification. Therefore, we can only estimate with what probability the given controller drives the distribution to the next collection of nodes without violating the specification. This allows us to construct a Markov decision process in which the states correspond to nodes, actions correspond to controller pairs, and transition probabilities correspond to the probability of the closed-loop system reaching the next node without violating the specification. Applying dynamic programming to this system yields a policy that maps the current region of belief states to the pair of controllers to be applied. Combining the policy with the synthesized LQG controllers yields a state-switched feedback controller that satisfies the system specifications with some minimum probability.

Given a Rabin automaton constructed from a GDTL formula and a FIRM, we construct a graph product between the two, called the GDTL-FIRM, to check if the state space has been sampled sufficiently to synthesize a switched controller satisfying the specification with positive probability. We use techniques similar to those in sampling-based formal synthesis work (Agha-mohammadi et al. 2014; Karaman and Frazzoli 2009, 2012; Vasile and Belta 2013, 2014) to construct the GDTL-FIRM incrementally until we find a policy with sufficiently high satisfaction probability.

1.1 Contributions

There are several contributions presented in this paper. First, we propose an end-to-end framework for motion planning with temporal and uncertainty goals supported by camera-equipped quadrotors. We also introduce Gaussian Distribution Temporal Logic, a fragment of distribution temporal logic that is restricted to properties involving the first and second moments of state distributions, that is used to formulate a control synthesis problem. Third, we present a sampling-based algorithm that leverages automata-based techniques, and Markov decision process (MDP) planning to generate satisfying control policies for stochastic ground vehicles. The functioning and performance of our framework are demonstrated experimentally.

Preliminary versions of parts of this work appear in Vasile et al. (2016) and Cristofalo et al. (2016). The novel elements of this work include the presentation of the entire end-to-end framework in a unified way with all details and proofs. We provide details on the conversion of GDTL to LTL for the construction of automata, proof of completeness, and an expanded discussion of the dynamic programming solution, as well as a comparison of localization with fixed camera networks versus a mobile camera mounted on a quadrotor.

1.2 Organization

The remainder of the paper is organized as follows. In Sec. 2, we formally introduce GDTL and state the problem under consideration. Our end-to-end solution method is presented in three parts. First, the mapping phase in Sec. 3. We propose a method for control synthesis in Sec. 4. The solution method concludes with the localization and tracking of the ground robot in Sec. 5. We present a case study and experimental results in Sec. 6, and our conclusions are given in Sec 7.

2 Problem Formulation

In this section, we define the problem of controlling a robot to satisfy a given GDTL formula with maximum probability.

Problem 1. Given a ground robot and a team of N quadrotors with on-board cameras operating in an unmapped environment, localize the ground robot, and design and execute a control policy to satisfy a GDTL formula ϕ with maximum probability.

Solving Problem 1 involves three main components for our robotic system, as shown in Fig. 1: first, a team of quadrotors constructs a mosaic map of the ground robot's operating environment, which is then labeled by a remote operator (Sec. 3). Next, a control policy is synthesized to satisfy the mission specification ϕ with maximum probability (Sec 4). To execute the control policy, a quadrotor localizes the ground robot using a downward facing camera (Sec. 5).

2.1 Motion and Observation Model

We consider an agent traveling in an environment according to dynamics

$$x_{k+1} = f\left(x_k, u_k, w_k\right), \tag{1}$$

where x_k is the state of the agent at time k, u_k is the input at time k, and w_k is the process noise at time k. The function $f\left(\cdot\right)$ is a locally Lipschitz continuous function representing the robot dynamics. The state and control space of the agent are given by $\mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{U} \subset \mathbb{R}^p$, respectively. For the moment, we place no assumptions on the distribution of the process noise w_k.

Noisy observations of the agent are given by the function

$$y_k = h\left(x_k, v_k\right), \tag{2}$$

where y_k is the observation at time k, v_k is the observation noise at time k, and $h\left(\cdot\right)$ is a locally Lipschitz continuous observation function. The observation space is $\mathcal{Y} \subset \mathbb{R}^m$. Again, we place no assumptions on the pdf of the observation noise v_k.

The state of the robot is estimated recursively using a Bayesian filter

$$b^{k+1} = \tau\left(b^k, u_k, y_{k+1}\right), \tag{3}$$

where $b^k = p(x_k \mid y_{1:k}, u_{0:k-1}) \in \mathbb{B}$ is a probability measure over the state space $\mathcal{X}$ at time k, representing the belief of the true state of the robot and $\mathbb{B}$ is the space of all such probability measures. The function $\tau\left(\cdot\right)$ is the update function for the filter. The prior belief at time 0 is given by b^0. The sequence of beliefs over time $b^0 b^1 b^2 \ldots$ is denoted by $\mathbf{b}$, and the suffix sequence $b^i b^{i+1} b^{i+2} \ldots$ is given by $\mathbf{b}^i$, $i \geq 0$.

2.2 Gaussian Distribution Temporal Logic

In this section, we define Gaussian Distribution Temporal Logic (GDTL), a predicate temporal logic defined over the space of Gaussian distributions with fixed dimension. GDTL is a fragment of distribution temporal logic (DTL) (Jones et al. 2013), but with the restriction that distributions be Gaussian. While we are able to synthesize control policies for GDTL, synthesizing control policies for full DTL is still a difficult open problem. DTL differs from other forms of probabilistic logic, e.g., quantitative LTL, Probabilistic Computational Tree Logic (CTL), and Probabilistic Signal Temporal Logic (STL), in that it is defined over sequences of distributions rather than over distributions of state sequences. This enables the definitions of predicates involving higher order moments and non-linear measures of uncertainty such as Shannon entropy or mutual information. In the case of Gaussian DTL, this corresponds to being able to specify how the covariance of our state estimate should behave over time. Further, GDTL can be used to express quantitative probability measures via inequalities involving the inverse Q function.

Let $\mathcal{G}$ denote the Gaussian belief space of dimension n, i.e. the space of Gaussian probability measures over $\mathbb{R}^n$. For brevity, we identify the Gaussian measures with their finite parametrization, mean and covariance matrix. Thus, $\mathcal{G} = \mathbb{R}^n \times S^n$. For a belief state $(x, P) \in \mathcal{G}$ we denote by $N_\delta(x, P) = \{b \in \mathcal{G} \mid \left\|b - (x, P)\right\|_{\mathcal{G}} \leq \delta\}$ the uncertainty ball of radius δ in the belief space centered at (x, P), where $\left\|\cdot\right\|_{\mathcal{G}}$ over $\mathcal{G}$ is a suitable norm in $\mathcal{G}$, e.g., $\left\|(x, P)\right\|_{\mathcal{G}} = \left\|x\right\|_2 + \alpha \left\|P\right\|_F$ with weight $\alpha > 0$.

Definition 1. GDTL Syntax. The *syntax* of Gaussian Distribution Temporal Logic is defined as

$$\phi := \top \mid g \leq 0 \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \mathcal{U} \phi_2,$$

where $\top$ is the Boolean constant "True", $g \leq 0$ is a predicate over $\mathcal{G}$, where $g : \mathcal{G} \to \mathbb{R}$, $\neg$ is negation ("Not"), $\wedge$ is conjunction ("And"), and $\mathcal{U}$ is "Until".

For convenience, we define the additional operators: $\phi_1 \vee \phi_2 \equiv \neg(\neg\phi_1 \wedge \neg\phi_2)$, $\Diamond\, \phi \equiv \top \mathcal{U} \phi$, and $\Box\, \phi \equiv \neg \Diamond \neg\phi$, where $\equiv$ denotes semantic equivalence.

Definition 2. GDTL Semantics. Let $\mathbf{b} = b^0 b^1 \ldots \in \mathcal{G}^\omega$ be an infinite sequence of belief states. The *semantics* of GDTL is defined recursively as

$$\begin{aligned}
&\mathbf{b}^i \models \top \\
&\mathbf{b}^i \models g \le 0 && \Leftrightarrow \quad g(b^i) \le 0 \\
&\mathbf{b}^i \models \neg\phi && \Leftrightarrow \quad \neg(\mathbf{b}^i \models \phi) \\
&\mathbf{b}^i \models \phi_1 \wedge \phi_2 && \Leftrightarrow \quad (\mathbf{b}^i \models \phi_1) \wedge (\mathbf{b}^i \models \phi_2) \\
&\mathbf{b}^i \models \phi_1 \vee \phi_2 && \Leftrightarrow \quad (\mathbf{b}^i \models \phi_1) \vee (\mathbf{b}^i \models \phi_2) \\
&\mathbf{b}^i \models \phi_1 \mathcal{U} \phi_2 && \Leftrightarrow \quad \exists j \ge i \text{ s.t. } (\mathbf{b}^j \models \phi_2) \\
& && \qquad \wedge (\mathbf{b}^k \models \phi_1, \forall k \in \{i, \ldots j-1\}) \\
&\mathbf{b}^i \models \Diamond\, \phi && \Leftrightarrow \quad \exists j \ge i \text{ s.t. } \mathbf{b}^j \models \phi \\
&\mathbf{b}^i \models \Box\, \phi && \Leftrightarrow \quad \forall j \ge i \text{ s.t. } \mathbf{b}^j \models \phi
\end{aligned}$$

The word $\mathbf{b}$ satisfies ϕ, denoted $\mathbf{b} \models \phi$, if and only if $\mathbf{b}^0 \models \phi$.

By allowing the definition of the atomic predicates used in GDTL to be quite general, we can potentially enforce interesting and relevant properties on the evolution of a system through belief space. Some of these properties include

- Bounds on $det(P)$, the determinant of covariance matrix P. This is used when we want to bound the overall uncertainty about the system's state.
- Bounds on $Tr(P)$, the trace of covariance matrix P. This is used when we want to bound the uncertainty about the system's state in any direction.
- Bounds on state mean $\hat{x}$. This is used when we want to specify where in state space the system should be.

Example 1. Let R be a robot evolving along a straight line with state denoted by $x \in \mathbb{R}$. The belief space for this particular robot is thus $(\hat{x}, P) \in \mathbb{R} \times [0, \infty)$, where $\hat{x}$ and P are its state estimate and covariance obtained from its sensors. The robot is tasked with going back and forth between two goal regions (denoted as $\pi_{g,1}$ and $\pi_{g,2}$ in the top of Fig. 2). It also must ensure that it never overshoots the goal regions or lands in obstacle regions $\pi_{o,1}$ and $\pi_{o,2}$. The robot must also maintain a covariance P of less than 0.5 m^2 at all times and less than 0.3 m^2 when in one of the goal regions. These requirements can be described by the GDTL formula

$$\begin{aligned}
\phi_{1d} &= \phi_{avoid} \wedge \phi_{reach} \wedge \phi_{u,1} \wedge \phi_{u,2}\text{ , where} \\
\phi_{avoid} &= \Box \neg((box(\hat{x}, -4, 0.35) \le 1) \\
&\qquad \vee (box(\hat{x}, 4, 0.35) \le 1)) \\
\phi_{reach} &= \Box \Diamond (box(\hat{x}, -2, 0.35) \le 1) \\
&\qquad \wedge \Box \Diamond (box(\hat{x}, 2, 0.35) \le 1) \\
\phi_{u,1} &= \Box (P < 0.5) \\
\phi_{u,2} &= \Box ((box(\hat{x}, -2, 0.35) \le 1) \\
&\qquad \wedge (box(\hat{x}, 2, 0.35) \le 1)) \Rightarrow (P < 0.3)\ ,
\end{aligned} \tag{4}$$

where $box(\hat{x}, x_c, a) = \left\| a^T (\hat{x} - x_c) \right\|_\infty$ is a function bounding $\hat{x}$ inside an interval of size $2|a|$ centered at x_c. Subformula ϕ_{avoid} encodes keeping the system away from the obstacle regions. Subformula ϕ_{reach} encodes periodically visiting the goal regions. Subformula $\phi_{u,1}$ encodes maintaining the uncertainty below 0.5 m^2 globally and subformula $\phi_{u,2}$ encodes maintaining the uncertainty below 0.3 m^2 in the goal regions.

The belief space associated with this problem is shown in the bottom of Fig. 2. The vertical lines in the figure correspond to the borders between the satisfaction and violation of predicates in (4), e.g. the level sets that are induced by the predicates when inequalities are replaced with equality. In the figure, + denotes that the predicate is satisfied in that region and - indicates that it is not. An example trajectory that satisfies (4) is shown with black dots[*]. Note that every point in this belief trajectory has covariance P less than 0.5, which satisfies $\phi_{u,1}$. Further, the forbidden regions in ϕ_{avoid} (marked with red stripes) are always avoided while each of the goal regions in ϕ_{reach} (marked with green stars) are each visited. Further, whenever the belief is in a goal region, it has covariance P less than 0.3, which means $\phi_{u,2}$ is satisfied.

2.3 GDTL synthesis problem definition

Solving Problem 1 requires the ability to synthesize a control policy that will satisfy a given GDTL specification. Here we formalize the problem of synthesizing such a policy:

Problem 2. GDTL Maximum Probability Problem. Let ϕ be a given GDTL formula and let the system evolve according to dynamics (1), with observation dynamics (2), and using Bayesian filter (3). Find a policy μ^* such that

$$\begin{aligned}
\mu^* = & \operatorname*{arg\,max}_{\mu \in \mathcal{M}(\mathfrak{S}, \mathcal{U})} Pr[\mathbf{b} \models \phi] \\
& \text{subject to (1), (2), (3),}
\end{aligned} \tag{5}$$

[*]Note that we consider a discrete time system in this example, and therefore the trajectory consists of a sequence of points in the state space. The black lines connecting those points only serve to clarify the sequence in which those states are visited.

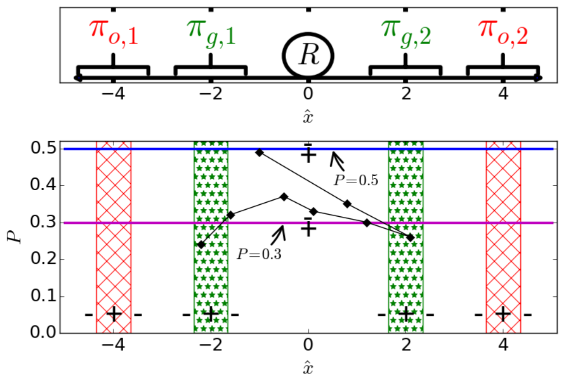


Figure 2. (Top) The state space of a system evolving along one dimension and (Bottom) the predicates from (4) as functions of the belief of the system from Ex. 1.

where $\mathcal{M}(\mathfrak{G}, \mathcal{U})$ denotes the admissible policies mapping beliefs in $\mathfrak{G}$ to control actions in $\mathcal{U}$.

Admissible policies must be deterministic and Borel measurable functions from $\mathfrak{G}$ to $\mathcal{U}$, otherwise the problem described by (5) is intractable. Thus, in our solution presented in Sec. 4 we restrict the space of policies similar to (Agha-mohammadi et al. 2014).

Example 2. We now present an example with a unicycle robot moving in a bounded planar environment (Fig. 3) to demonstrate a realistic GDTL specification. The specification is given over belief states associated with the measurement y of the robot as follows: "Visit regions A and B infinitely many times. If region A is visited, then only corridor D_1 may be used to cross to the right side of the environment. Similarly, if region B is visited, then only corridor D_2 may be used to cross to the left side of the environment. The obstacle Obs in the center must always be avoided. The uncertainty must always be less than $0.9m^2$. When passing through the corridors D_1 and D_2 the uncertainty must be at most $0.6m^2$."

The corresponding GDTL formula is:

$$\begin{aligned}
\phi_1 &= \phi_{avoid} \wedge \phi_{reach} \wedge \phi_{u,1} \wedge \phi_{u,2} \wedge \phi_{bounds} \qquad (6)\\
\phi_{avoid} &= \Box\, \neg\phi_{Obs}\\
\phi_{reach} &= \Box\, \big(\Diamond\, (\phi_A \wedge \neg\phi_{D_2} \mathcal{U} \phi_B) \wedge \Diamond\, (\phi_B \wedge \neg\phi_{D_1} \mathcal{U} \phi_A)\big)\\
\phi_{u,1} &= \Box\, (tr(P) \leq 0.9)\\
\phi_{u,2} &= \Box\, \big((\phi_{D_1} \vee \phi_{D_2}) \Rightarrow (tr(P) \leq 0.6)\big)\\
\phi_{bounds} &= \Box\, (box(\hat{x}, x_c, a) \leq 1),
\end{aligned}$$

where $(\hat{x}, P)$ is a belief state associated with y, $a = \begin{bmatrix} \frac{2}{l} & \frac{2}{w} & 0 \end{bmatrix}$ so that $\hat{x}$ must remain within a rectangular $l \times w$ region with center $x_c = \begin{bmatrix} \frac{l}{2} & \frac{w}{2} & 0 \end{bmatrix}$, $l = 4.13\,m$ and $w = 3.54\,m$. The 5 regions in the environment are defined by GDTL predicate formulae

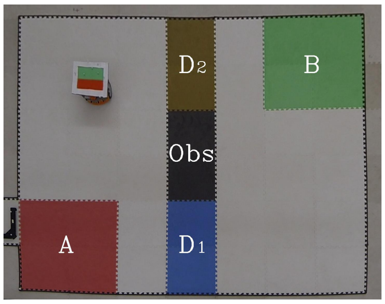


Figure 3. Environment with two regions A and B, two corridors D_1 and D_2 and an obstacle Obs.

$\phi_{Reg} = (box(\hat{x}, x_{Reg}, r_{Reg}) \leq 1)$, where x_{Reg} and r_{Reg} are the center and the dimensions of region $Reg \in \{A, B, D_1, D_2, Obs\}$, respectively.

3 Map building

Solving Problem 1 requires knowledge of the operating environment for the ground robot. The environment must therefore be mapped so the ground robot can satisfy its mission. This section serves to explain the process of building a map of the environment with a team of quadrotors equipped with cameras. It is worth noting that this component of our solution framework could be omitted if given an appropriate high resolution map, such as a satellite image. Nonetheless, mapping may be necessary in cases for which satellite imagery doesn't match recent changes to the environment, such as following a natural disaster. Likewise, normal environmental variations, such as seasonal vegetation and snow cover changes, moving vehicles in urban areas, and moving equipment in industrial parks may vary substantially from a satellite image.

3.1 Inter-Image Homography

Map building and ground robot pose estimation rely on the inter-image homography, $\mathbf{H}_{ij} \in \mathbb{R}^{3\times3}$, which defines the linear transformation between co-planar three-dimensional (3D) points described in two different coordinate frames, i.e., $\mathbf{P}_i = \mathbf{H}_{ij}\mathbf{P}_j$, where $\mathbf{P}_i \in \mathbb{R}^3$ and $\mathbf{P}_j \in \mathbb{R}^3$. The perspective projection of these 3D points yields the measured image features, $\mathbf{p}_i \in \mathbb{R}^2$ and $\mathbf{p}_j \in \mathbb{R}^2$, that are given by the cameras i and j, respectively. These two image features are related by the following homography, $\mathbf{p}_i = \tilde{\mathbf{H}}_{ij}\mathbf{p}_j$, where $\tilde{\mathbf{H}}_{ij} = \mathbf{K}\mathbf{H}_{ij}\mathbf{K}^{-1}$ is estimated using standard least squares estimation (Ma et al. 2004) with at least four matched image feature points, and $\mathbf{K}$ is the known calibration matrix of the identical cameras. In this work, we assume that all quadrotors are flying at a sufficiently high altitude to justify the co-planar requirements of points on the ground. The *rectified* homography describes the transformation between two parallel, calibrated camera poses,

$$\mathbf{H}^r_{ij} = \begin{bmatrix} \cos(\psi_{ij}) & -\sin(\psi_{ij}) & -\frac{x_{ij}}{z_j} \\ \sin(\psi_{ij}) & \cos(\psi_{ij}) & -\frac{y_{ij}}{z_j} \\ 0 & 0 & 1-\frac{z_{ij}}{z_j} \end{bmatrix}, \quad (7)$$

where $[x_{ij}, y_{ij}, z_{ij}, \psi_{ij}]^T \in \mathbb{R}^4$ is the estimated parallel pose of camera j in the frame of camera i. We remove the roll and pitch effect of a translating quadrotor from the acquired image, i.e., $\mathbf{H}^r_{ij} = \mathbf{R}_{\theta_i}\mathbf{R}_{\phi_i}\mathbf{K}^{-1}\tilde{\mathbf{H}}_{ij}\mathbf{K}\mathbf{R}^T_{\phi_j}\mathbf{R}^T_{\theta_j}$, given the roll, ϕ, and pitch, θ, of each quadrotor. We extract the relative position from the last column of $\mathbf{H}^r_{ij}$, given the altitude of the cameras provided by the altimeter, and the relative orientation from the upper 2×2 block of $\mathbf{H}^r_{ij}$.

3.2 Homography-based Formation Control

Homography-based formation control (Montijano et al. 2016) drives the team of quadrotors that generates the high fidelity mosaic map image, which is a composite image of the quadrotors' onboard images while in formation. The consensus-based kinematic control laws that drive the formation of quadrotors to their desired relative pose, $[x^*_{i,j}, y^*_{i,j}, \psi^*_{i,j}]^T$, are functions of the computed rectified homography from equation (7), i.e.,

$$w_{z_i} = K_w \sum_{j\in\mathcal{N}_i} \left(\arctan\left[\frac{[\mathbf{H}^r_{ij}]_{21}}{[\mathbf{H}^r_{ij}]_{11}}\right] - \psi^*_{i,j} \right), \quad (8)$$

$$\begin{bmatrix} v_{x_i} \\ v_{y_i} \end{bmatrix} = K_v \sum_{j\in\mathcal{N}_i} \left(\begin{bmatrix} [\mathbf{H}^r_{ij}]_{13} \\ [\mathbf{H}^r_{ij}]_{23} \end{bmatrix} - \begin{bmatrix} x^*_{i,j} \\ y^*_{i,j} \end{bmatrix} \right), \quad (9)$$

$$v_{z_i} = K_v \sum_{j\in\mathcal{N}_i} \left(1 - [\mathbf{H}^r_{ij}]_{33}\right), \quad (10)$$

where $[v_{x_i}, v_{y_i}, v_{z_i}]^T$ is the translational velocity control and w_{z_i} is the rotational velocity control about the z-axis of the quadrotor, i.e., its yaw. Note that the element in row a and column b of $\mathbf{H}^r_{ij}$ is denoted by $[\mathbf{H}^r_{ij}]_{ab}$. The relative yaw does not affect z_{ij}, therefore, the relative altitude can be controlled towards zero using $[\mathbf{H}^r_{ij}]_{33}$. The team produces the mosiac map of the environment when the quadrotors reach the chosen formation that yields sufficient image overlap for accurate pose estimation and large enough field of view to cover the region of interest in the environment. It is worth noting that this component of our solution framework could be omitted if a high resolution map is already available, such as from a satellite image.

With the map constructed , we can synthesize a control policy that satisfies a GDTL mission specification over the regions in the map (Sec. 4). In this work, we assume that these regions are either remotely labeled by a human operator or that semantic segmentation of the map allows the agents themselves to recognize relevant regions of interest in the map. Incorporating semantic segmentation into our framework is an interesting direction for future work, but is outside the scope of this paper. Semantic segmentation and labeling have their own body of literature (Simonyan and Zisserman 2014; Russakovsky et al. 2015).

4 Control Policy Synthesis

In this section, we present the details for synthesis of a control policy for satisfying a GDTL specification. We use a sampling-based algorithm to turn the continuous environment into a discrete graph. The nodes in the graph are beliefs, and the edges are transitions between beliefs. The transitions correspond to the robot acting under a feedback controller that drives the robot from one node to the other. We compute the feedback controllers associated with transitions by liniarizing our system around the destination node. Thus the robot moving in the graph can be represented as a probabilistic transition system, allowing us to apply existing tools from formal synthesis to obtain a control policy. Following standard methods, we combine the probabilistic transition system with the specification to form a product MDP, which is ultimately used to find a policy satisfying the specification with highest probability. This policy is implemented on the robot as a sequence of switching feedback controllers applied along different edges of the graph, driving the robot to maximize the probability of satisfying its specification in the continuous environment.

4.1 Linearization

We begin by linearizing our system in order to compute an approximately optimal LQG controller that can be used by our sampling-based algorithm (Sec. 4.2). Thus, the system should have noisy linear time-invariant (LTI) dynamics around randomly sampled states given by

$$x_{k+1} = Ax_k + Bu_k + w_k, \quad (11)$$

where $x_k \in \mathcal{X}$ is the state of the system, $\mathcal{X} \subseteq \mathbb{R}^n$ is the state space, $A \in \mathbb{R}^{n\times n}$ is the dynamics matrix, $B \in \mathbb{R}^{n\times p}$ is the control matrix, $u_k \in \mathcal{U}$ is a control signal, $\mathcal{U} \subseteq \mathbb{R}^p$ is the control space, and w_k is a zero-mean Gaussian process with covariance $Q \in \mathbb{R}^{n\times n}$. Example 2 below demonstrates how this model can be applied to our unicycle robot.

The state is observed indirectly according to the linear observation model

$$y_k = Cx_k + v_k, \quad (12)$$

where $y_k \in \mathcal{Y}$ is a measurement, $\mathcal{Y} \subseteq \mathbb{R}^m$ is the observation space, $C \in \mathbb{R}^{m\times n}$ is the observation matrix and v_k is a zero-mean Gaussian process with covariance $R \in \mathbb{R}^{m \times m}$. We note that the noise in the pixel space may not be Gaussian for more accurate lens models or other measurement models for that matter. Nevertheless, the additive Gaussian noise assumption is generally sufficient for most robotics applications, including nonlinear pinhole cameras used for EKF-SLAM (Davison et al. 2007) or target tracking with mobile cameras (Li and Jilkov 2001; Campbell and Whitacre 2007; Cognetti et al. 2018).

We require that the LTI system (11), (12) be controllable and observable, i.e., (A, B) is a controllable pair and (A, C) is an observable pair. Moreover, we require that C be full rank. These requirements apply to many systems, including nonlinear systems that can be linearized to satisfy the assumptions. This is demonstrated in example 2 below.

Given the LTI system described above, the belief state at each time step is characterized by the *a posteriori* state and error covariance estimates, $\hat{x}_k$ and P_k, i.e., $b^k = (\hat{x}_k, P_k) \in \mathbb{B}$, where $\mathbb{B}$ is the space of probability measures over the state space $\mathcal{X}$ at time k. The belief state is maintained via a Kalman filter (Kalman 1960), which we denote compactly as

$$b^{k+1} = \tau(b^k, u_k, y_{k+1}), \;\; b^0 = (\hat{x}_0, P_0) \,, \qquad (13)$$

where b^0 is the known initial belief about the system's state centered at $\hat{x}_0$ with covariance P_0. The sequence of beliefs over time $b^0 b^1 b^2 \ldots$ is denoted $\mathbf{b}$, and the suffix sequence $b^i b^{i+1} b^{i+2} \ldots$ is given by $\mathbf{b}^i$, $i \geq 0$.

Note that linearization introduces approximation errors that have an impact both on the controllers and filters. Both problems have been extensively studied in the literature, and are outside the scope of this paper. A detailed discussion can be found in (Agha-mohammadi et al. 2014) (in the context of FIRM) and references therein.

Example 2. *Continued.* We continue our running example of a unicycle robot moving in the environment shown in Fig. 3. This robot model is non-linear, but we can approximate the robot's dynamics using LTI systems with Gaussian noise around samples in the workspace. This heuristic is very common, since the non-linear and non-Gaussian cases yield recursive filters that do not in general admit finite parametrization. We first discretize the system dynamics using Euler's approximation. The motion model becomes:

$$x_{k+1} = f(x_k, u_k, w_k) = x_k + \begin{bmatrix} \cos(\theta_k) & 0 \\ \sin(\theta_k) & 0 \\ 0 & 1 \end{bmatrix} \cdot u_k + w_k \quad (14)$$

where $x_k = \left[p^x_k \; p^y_k \; \theta_k\right]^T$, p^x_k, p^y_k and θ_k are the position and orientation of the robot in a global reference frame, $u_k = \left[v'_k \; \omega'_k\right]^T = \Delta t \left[v_k \; \omega_k\right]^T$, v_k and ω_k are the linear and rotation velocities of the robot, Δt is the discretization step, and w_k is a zero-mean Gaussian process with covariance matrix $Q \in \mathbb{R}^{3\times 3}$. Next, we linearize the system around a nominal operating point (x^d, u^d) without noise,

$$x_{k+1} = f(x^d, u^d, 0) + A\,(x_k - x^d) + B\,(u_k - u^d) + w_k, \qquad (15)$$

where $A = \frac{\partial f}{\partial x_k}(x^d, u^d, 0)$ and $B = \frac{\partial f}{\partial u_k}(x^d, u^d, 0)$ are the process and control Jacobians, $x^d = \left[p^{x\,d} \; p^{y\,d} \; \theta^d\right]^T$, and $u^d = \left[v'^d_k \; \omega'^d_k\right]^T$. In our framework, we associate with each belief node B_g centered at $(\hat{x}^g, P)$ an LTI system obtained by linearization (15) about $(\hat{x}^g, u^g)$, where $u^g = [0.1, \, 0]^T$ corresponds to 0.1 m/s linear velocity and 0 angular velocity.

Continuing our example, we show how the linear observation model can be achieved, localizing the robot with a multiple camera network. This model reflects the real world constraints of sensor networks, e.g. finite coverage, finite resolution, and improved accuracy with the addition of more sensors. Likewise, a linear observation model can be achieved using cameras (or a network of cameras as in Fig. 5(a)). The estimation of the planar position and orientation of the robot in the global frame is formulated as a least squares problem (*structure from motion*) (Ma et al. 2004). The measurement, $y_k \in \mathcal{Y}$, is given by the discrete observation model: $y_k = Cx_k + v_k$. The measurement error covariance matrix is defined as $R = \mathrm{diag}(r_x, r_y, r_\theta)$, where the value of each scalar is inversely proportional to the number of cameras used in the estimation, i.e. the number of camera views that identify the robot. These values are generated from a camera coverage map (Fig. 5(a)) of the experimental space.

4.2 Sampling-based algorithm

We propose a sampling-based algorithm to solve Pb. 2 that overcomes the curse of dimensionality and history generally associated with POMDPs. In short, a sampling-based algorithm iteratively grows a graph $\mathcal{T}$ in the state space, where nodes are individual states, and edges correspond to motion primitives that drive the system from state to state (LaValle 2006). The extension procedure is biased towards exploration of uncovered regions of the state space. Similar to (Agha-mohammadi et al. 2014), we adapt sampling-based methods to produce finite abstractions (e.g., graphs) of the belief space.

Alg. 1 incrementally constructs a transition system $\mathcal{T} = (\mathfrak{B}_\mathcal{T}, B_0, \Delta_\mathcal{T}, \mathcal{C}_\mathcal{T})$, where the state space $\mathfrak{B}_\mathcal{T}$ is composed of belief nodes, i.e., bounded hyper-balls in $\mathcal{G}$, $\Delta_\mathcal{T}$ is the set of transitions, and $\mathcal{C}_\mathcal{T}$ is a set of controllers associated with edges. The center of a belief node is a belief state $b = (x, P^\infty)$, where the mean x is obtained through random sampling of the system's linearized state space, and P^∞ is the stationary covariance. The initial belief node is denoted by B_0.

Sampling-based algorithms are built using a set of primitive functions that are assumed to be available:

- $sample(\mathcal{X})$ generates random states from a distribution over the state space $\mathcal{X}$,
- $nearest(x^r, \mathcal{T}) = \arg\min_{x^u} \{\|x^r - x^u\|_2 \mid \exists P^u \wedge N_\delta(x^u, P^u) \in \mathfrak{B}_\mathcal{T}\}$ returns the mean x^u of a belief node's center in $\mathcal{T}$ such that x^u is closest to the state x^r using the metric defined on $\mathcal{X}$,
- $near(B_n, \mathfrak{B}_\mathcal{T}, \gamma)$ returns the closest γ belief nodes in $\mathfrak{B}_\mathcal{T}$ to B_n with respect to the distance between them induced by $\|\cdot\|_\mathcal{G}$, and
- $steer(x^i, x^t)$ returns a state obtained by attempting to drive the system from x^i towards x^t.

Note that the $near$ function uses the distance $\|\cdot\|_\mathcal{G}$ as defined in Section 2.2, which can incorporate the means and covariance of the belief nodes. Using these primitive functions, an extension procedure $extend(\mathcal{X}, \mathcal{T})$ of the transition system $\mathcal{T}$ can be defined as:

1. generate a new sample $x^r \leftarrow sample(\mathcal{X})$,
2. find nearest state $x^u \leftarrow nearest(x^r, \mathcal{T})$, and
3. drive the system towards the random sample $x^n \leftarrow steer(x^u, x^r)$.

For more details about sampling-based algorithms, primitive functions and their implementations see (LaValle 2006; Karaman and Frazzoli 2011; Vasile and Belta 2013).

Transitions are enforced using local controllers which are stored in $\mathcal{C}_\mathcal{T}$. i.e., we assign to each edge $e \in \Delta_\mathcal{T}$ a local controller $ec_e \in \mathcal{C}_\mathcal{T}$. The local controllers take the form of LQG controllers as described in Section 4.1, and are used to enforce computed transitions to states generated using the nominal robot dynamics within the $steer$ function. Under the assumptions of our model (Agha-mohammadi et al. 2014), the local controllers are guaranteed to stabilize the system to belief nodes along a path in finite time. Thus we abstract the roadmap to a deterministic system. In Alg. 1, local controllers are generated using the method $localController()$. The design of the node controllers is presented Sec. 6.2.

Example 2. *Continued.* Our running example continues, demonstrating the design of the node controllers. We used the following simple switching controller to drive the robot towards belief nodes:

$$u_{k+1} = \begin{cases} \left[k_D \left\|\alpha^T(x^g - \hat{x}_k)\right\|_2 \quad k_\theta(\theta_k^{los} - \hat{\theta}_k)\right]^T & \text{if } \left|\theta_k^{los} - \hat{\theta}_k\right| < \frac{\pi}{12} \\ \left[0 \quad k_\theta(\theta_k^{los} - \hat{\theta}_k)\right]^T, & \text{otherwise} \end{cases},$$

where $k_D > 0$ and $k_\theta > 0$ are proportional scalar gains, x^g is the goal position, θ_k^{los} is the line-of-sight angle and $\alpha = [1\ 1\ 0]^T$. We assume, as in (Agha-mohammadi et al. 2014), that the controller is able to stabilize the system state and uncertainty around the goal belief state (x^g, P^∞), where P^∞ is the stationary covariance matrix.

The predicates considered in GDTL are defined over the belief space $\mathcal{G}$ and describe sets in this space. However, because the uncertainty is separate from the temporal ordering of the satisfaction of the predicates, the sets determined by the predicates are independent of the position of a belief state b^i in a belief word $\mathbf{b}$. As a consequence, we can convert a GDTL formula into an LTL formula.

Denote $\mathcal{G}_f = \{b \in \mathcal{G} \mid f(b) \leq 0\}$, where $f \in \mathcal{F}(\mathcal{G}, \mathbb{R})$, and the set of predicates in GDTL formula ϕ as F_ϕ.

Definition 3. LTL Equivalent. Let ϕ be a GDTL formula and F_ϕ be the set of all predicates in ϕ. Let AP be a finite set such that $|AP| = |F_\phi|$ and a bijective map $\widetilde{\ }: F_\phi \to AP$. Consider the LTL formula φ, where each predicate in F_ϕ is substituted by its associated atomic proposition in AP using the map $\widetilde{\ }$. The semantics of φ are given with respect to infinite words in $\mathcal{G}^\omega$. Satisfaction of an atomic proposition $\mathbf{b} \models \widetilde{p}$ is interpreted as $b^0 \in \mathcal{G}_f$, where $p = (f \leq 0) \in F_\phi$. The Boolean and temporal operators retain their usual meaning.

There exist efficient algorithms that translate LTL formulae into Rabin automata (Klein and Baier 2006). The algorithm checks for the presence of a satisfying path using a deterministic Rabin automaton (DRA) $\mathcal{R}$ that is computed from the GDTL specification.

Definition 4. Rabin Automaton. A (deterministic) Rabin automaton is a tuple $\mathcal{R} = (S_\mathcal{R}, s_0^\mathcal{R}, \Sigma, \delta, \Omega_\mathcal{R})$, where $S_\mathcal{R}$ is a finite set of states, $s_0^\mathcal{R} \in S_\mathcal{R}$ is the initial state, $\Sigma \subseteq 2^{F_\phi}$ is the input alphabet, $\delta : S_\mathcal{R} \times \Sigma \to S_\mathcal{R}$ is the transition function, and $\Omega_\mathcal{R}$ is a set of tuples $(\mathcal{F}_i, \mathcal{B}_i)$ of disjoint subsets of $S_\mathcal{R}$ which correspond to good ($\mathcal{F}_i$) and bad ($\mathcal{B}_i$) states.

A transition $s' = \delta(s, \sigma)$ is also denoted by $s \xrightarrow{\sigma}_\mathcal{R} s'$. A trajectory of the Rabin automaton $\mathbf{s} = s_0 s_1 \ldots$ is generated by an infinite sequence of symbols $\boldsymbol{\sigma} = \sigma_0\sigma_1 \ldots$ if $s_0 = s_0^\mathcal{R}$ is the initial state of $\mathcal{R}$ and $s_k \xrightarrow{\sigma_k}_\mathcal{R} s_{k+1}$ for all $k \geq 0$. Given a state trajectory $\mathbf{s}$ we define $\vartheta_\infty(\mathbf{s}) \subseteq S_\mathcal{R}$ as the set of states which appear infinitely many times in $\mathbf{s}$. An infinite input sequence over Σ is said to be accepted by a Rabin automaton $\mathcal{R}$ if there exists a tuple $(\mathcal{F}_i, \mathcal{B}_i) \in \Omega_\mathcal{R}$ of good and bad states such that the state trajectory $\mathbf{s}$ of $\mathcal{R}$ generated by $\boldsymbol{\sigma}$ intersects the set $\mathcal{F}_i$ infinitely many times and the set $\mathcal{B}_i$ only finitely many times. Formally, this means that $\vartheta_\infty(\mathbf{s}) \cap \mathcal{F}_i \neq \emptyset$ and $\vartheta_\infty(\mathbf{s}) \cap \mathcal{B}_i = \emptyset$.

4.3 Computing transition and intersection probability

At this point in our solution, we have a framework for generating a DRA and a *deterministic* transition system in the belief space. Nonetheless, their interaction results in

Algorithm 1: $ConstructTS(x_0, \phi, \varepsilon)$

```
Input: initial state $x^0$, GDTL specification $\phi$, and
       lower bound $\varepsilon$
Output: belief transition system $\mathcal{T}$, product MDP $\mathcal{P}$,
        and satisfying policy $\mu^*$
1  convert GDTL formula $\phi$ to LTL formula $\varphi$ over the
   set of atomic propositions $AP = F_\phi$
2  compute DRA $\mathcal{R} = (S_\mathcal{R}, s_0^\mathcal{R}, 2^{AP}, \delta, \Omega_\mathcal{R})$ from $\varphi$
3  $ec_0, P_0^\infty \leftarrow localController(x^0)$
4  $B_0 \leftarrow N_\delta(x^0, P_0^\infty)$
5  $e_0 = (B_0, B_0)$
6  $\pi_0^{S_\mathcal{R}}, \pi_0^{\Omega_\mathcal{R}} \leftarrow computeProb(e_0, s_0, ec_0, \mathcal{R})$
7  initialize belief TS $\mathcal{T} = (\mathfrak{B}_\mathcal{T} = \{B_0\}, B_0, \Delta_\mathcal{T} =$
   $\{e_0\}, \mathcal{C}_\mathcal{T} = \{(e_0, ec_0)\})$
8  construct product MDP
   $\mathcal{P} = \mathcal{T} \times \mathcal{R} = (S_\mathcal{P} = \mathfrak{B}_\mathcal{T} \times S_\mathcal{R}, (B_0, s_0), Act =$
   $\mathfrak{B}_\mathcal{T}, \delta_\mathcal{P} = \{\pi_0^{S_\mathcal{R}}\}, \Omega_\mathcal{P} = \{\pi_0^{\Omega_\mathcal{R}}\})$
9  for $index = 1$ to $N$ do
10     $x^n \leftarrow extend(\mathcal{X}, \mathcal{T})$
11     $ec_n, P_n^\infty \leftarrow localController(x^n)$
12     $B_n \leftarrow N_\delta(x^n, P_n^\infty)$
13     $\mathcal{N}_n \leftarrow near(B_n, \mathfrak{B}_\mathcal{T}, \gamma)$
14     $\Delta_n \leftarrow \{(B_i, B_n) | x^n = steer(x^i, x^n), B_i \in \mathcal{N}_n\}$
           $\cup \{(B_n, B_i) | x^i = steer(x^n, x^i), B_i \in \mathcal{N}_n\}$
15     $\mathfrak{B}_\mathcal{T} \leftarrow \mathfrak{B}_\mathcal{T} \cup \{B_n\}, \Delta_\mathcal{T} \leftarrow \Delta_\mathcal{T} \cup \Delta_n$
16     $S_\mathcal{P} \leftarrow S_\mathcal{P} \cup (\{B_n\} \times S_\mathcal{R})$
17     foreach $e = (B_u, B_v) \in \Delta_n$ do
18         $\mathcal{C}_\mathcal{T} \leftarrow \mathcal{C}_\mathcal{T} \cup \{(e, ec_v)\}$
19         foreach $s_u \in S_\mathcal{R}$ s.t. $(B_u, s_u) \in S_\mathcal{P}$ do
20             $\pi_e^{S_\mathcal{R}}, \pi_e^{\Omega_\mathcal{R}} \leftarrow computeProb(e, s_u, ec_v, \mathcal{R})$
21             $\delta_\mathcal{P} \leftarrow \delta_\mathcal{P} \cup \{\pi_e^{S_\mathcal{R}}\}$
22             $\Omega_\mathcal{P} \leftarrow \Omega_\mathcal{P} \cup \{\pi_e^{\Omega_\mathcal{R}}\}$
23     $\Delta_\mathcal{P}^n = \{(p, p') \in \Delta_\mathcal{P} \mid (p, p')\upharpoonright_\mathcal{T} \in \Delta_n\}$
24     foreach $(\mathcal{F}_i, \mathcal{B}_i) \in \Omega_\mathcal{R}$ do // update ECs
25         $\Gamma_i = \{(p, p') \in \Delta_\mathcal{P}^n \mid \pi^{\Omega_\mathcal{R}}(e, \mathcal{F}_i) = 0$
               $\wedge \pi^{\Omega_\mathcal{R}}(e, \mathcal{B}_i) > 0, e = (p, p')\upharpoonright_\mathcal{T}\}$
26         $c_i.update(\Delta_\mathcal{P}^n \setminus \Gamma_i)$
27     if $existsSatPolicy(\mathcal{P})$ then
28         solve DP (18) and compute policy $\mu^*$ with
           probability of satisfaction $p$
29         if $p \geq \varepsilon$ then return $(\mathcal{T}, \mathcal{P}, \mu^*)$
30 return $(\mathcal{T}, \mathcal{P}, \emptyset)$
```

a *probabilistic* MDP, as the noisy motion and observation models induce a distribution of possible trajectories in the DRA as the system travels between one belief node and another. In this section we describe how to compute these probabilities.

Given a transition $e = (B_u, B_v)$ and a local controller ec_e, we wish to compute the associated transition in the DRA. However the transition between belief nodes B_u and B_v depends on the motion and observation noise. Thus, there exists a distribution of possible trajectories between those two belief nodes. This in turn means there is a distribution over trajectories in the DRA, including with respect to the acceptance condition. It is unclear how to characterize and estimate these distributions directly. Therefore we approximate these distributions using multiple trajectory samples of the closed-loop system enforcing edge e. Instead of estimating distributions of DRA trajectories, we compute (marginal) distributions, over terminal DRA states and over intersection with good and bad states and neither. Alg. 2 computes the transition distribution from a start DRA state s_u to some random DRA state, and a set of intersection distributions associated with each pair $(\mathcal{F}_i, \mathcal{B}_i)$ of the acceptance set of $\mathcal{R}$. In Alg. 2, the function $sampleBeliefSet(S)$ returns a random sample from a uniform distribution over the belief set S.

Algorithm 2: $computeProb(e = (B_u, B_v), s_u, ec_e, \mathcal{R})$

```
Input: transition between belief nodes $e = (B_u, B_v)$,
       starting DRA state $s_u$, controller enforcing $e$
       $ec_e$, and deterministic Rabin automaton $\mathcal{R}$
Output: transition distribution $\pi^{S_\mathcal{R}}$, and intersection
        distribution $\pi^{\Omega_\mathcal{R}}$
Parameter: $NP$ – number of particles
1  $t \leftarrow \mathbf{0}_{|S_\mathcal{R}|,1}$
2  $ra_i \leftarrow \mathbf{0}_{3,1}, \forall (\mathcal{F}_i, \mathcal{B}_i) \in \Omega_\mathcal{R}$
3  for $p = 1 : NP$ do
4      $b_u \leftarrow sampleBeliefSet(B_u)$
5      $b^{0:T} \leftarrow ec_e(b_u)$
6      for $k = 0$ to $T - 1$ do
7          $\sigma_k \leftarrow \{f \mid f(b^k) \leq 0, \forall f \in F_\phi\}$
8      $\mathbf{s} = s_{0:T} \leftarrow (s_u \overset{\sigma_{0:T-1}}{\rightarrow} s_T)$
9      $t[s_T] \leftarrow t[s_T] + 1$
10     for $(\mathcal{F}_i, \mathcal{B}_i) \in |\Omega_\mathcal{R}|$ do
11         if $\mathcal{F}_i \cap \mathbf{s} \neq \emptyset$ then $ra_i[1] \leftarrow ra_i[1] + 1$
12         if $\mathcal{B}_i \cap \mathbf{s} \neq \emptyset$ then $ra_i[2] \leftarrow ra_i[2] + 1$
13         if $(\mathcal{F}_i \cup \mathcal{B}_i) \cap \mathbf{s} = \emptyset$ then $ra_i[3] \leftarrow ra_i[3] + 1$
14 return $\left(\pi^{S_\mathcal{R}} = \frac{t}{NP}, \pi^{\Omega_\mathcal{R}} = \left\{\frac{ra_i}{NP} \mid 1 \leq i \leq |\Omega_\mathcal{R}|\right\}\right)$
```

The distributions we estimate in the DRA $\mathcal{R}$ are written formally as follows. The distribution $\pi^{S_\mathcal{R}}$ captures the probability of the terminal DRA state s_v obtained by executing controller ec_e to drive the system from belief node B_u to belief node B_v starting with DRA state s_u. That is, $\pi^{S_\mathcal{R}} = Pr[s_v \mid e, s_u, ec_e]$, where $s_v \in S_\mathcal{R}$ is the terminal DRA state, $s_u \overset{\sigma_{0:T-1}}{\rightarrow} s_v$ is the (random) DRA trajectory, $b^{0:T} = ec_e(b_u)$, $b_u \in B_u$ is the (random) belief trajectory, and $\sigma_k \leftarrow \{f \mid f(b^k) \leq 0, \forall f \in F_\phi\}$ is the (random) output word (the sequence of predicate sets that evaluate true at each step).

We also introduce distributions to keep track of satisfaction and violation along transitions, which we call intersections distributions. Each intersection distribution represents the probability that a DRA trajectory induced by edge e intersects $\mathcal{F}_i$, $\mathcal{B}_i$ or neither, where $(\mathcal{F}_i, \mathcal{B}_i) \in \Omega_\mathcal{R}$, and the controller ec_e was used to drive the system along the edge e starting from the DRA state s_u:

$$\pi^{\Omega_\mathcal{R}} = \left\{ \begin{array}{l|l} Pr[\mathbf{s} \cap \mathcal{F}_i \mid e, s_u, ec_e] & \\ Pr[\mathbf{s} \cap \mathcal{B}_i \mid e, s_u, ec_e] & \forall(\mathcal{F}_i, \mathcal{B}_i) \in \Omega_\mathcal{R} \\ Pr[\mathbf{s} \cap (\mathcal{F}_i \cup \mathcal{B}_i) \mid e, s_u, ec_e] & \end{array} \right\} \tag{16}$$

For convenience, we use the following notation $\pi^{\Omega_\mathcal{R}}(e, X) = Pr[\mathbf{s} \cap X \mid e, s_u, ec_e]$, where $X \in \{\mathcal{F}_i, \mathcal{B}_i, \mathcal{F}_i \cup \mathcal{B}_i\}$.

4.4 GDTL-FIRM Product MDP

In this section, we define a construction procedure of the product MDP between the (belief) TS $\mathcal{T}$ and the specification DRA $\mathcal{R}$.

Definition 5. GDTL-FIRM MDP. Given a DTS $\mathcal{T} = (\mathfrak{B}_\mathcal{T}, B_0, \Delta_\mathcal{T}, \mathcal{C}_\mathcal{T})$, a Rabin automaton $\mathcal{R} = (S_\mathcal{R}, s_0^\mathcal{R}, \Sigma = 2^{AP}, \delta, \Omega_\mathcal{R})$, and the transition and intersection probabilities $\pi^{S_\mathcal{R}}$, $\pi^{\Omega_\mathcal{R}}$, their product MDP, denoted by $\mathcal{P} = \mathcal{T} \times \mathcal{R}$, is a tuple $\mathcal{P} = (S_\mathcal{P}, s_0^\mathcal{P}, Act, \delta_\mathcal{P}, \Omega_\mathcal{P})$ where $s_0^\mathcal{P} = (B_0, s_0^\mathcal{R})$ is the initial state; $S_\mathcal{P} \subseteq \mathfrak{B}_\mathcal{T} \times S_\mathcal{R}$ is a finite set of states which are reachable from the initial state by run of positive probability (see below); $Act = \mathfrak{B}_\mathcal{T}$ is the set of actions available at each state; $\delta_\mathcal{P} : S_\mathcal{P} \times Act \times S_\mathcal{P} \to [0, 1]$ is the transition probability defined by $\delta_\mathcal{P}((B_i, s_i), B_j, (B_j, s_j)) = \pi^{S_\mathcal{R}}(s_j; e_{ij}, s_i, \mathcal{C}_\mathcal{T}(e_{ij}))$, $e_{ij} = (B_i, B_j)$; and $\Omega_\mathcal{P}$ is the set of tuples of good and bad transitions in the product automaton.

Denote the set of edges of positive probability by $\Delta_\mathcal{P} = \{((B_i, s_i), (B_j, s_j)) \mid \delta_\mathcal{P}((B_i, s_i), B_j, (B_j, s_j)) > 0\}$. A transition in $\mathcal{P}$ is also denoted by $p_i \to_\mathcal{P} p_j$ if $(p_i, p_j) \in \Delta_\mathcal{P}$. A trajectory (or run) of *positive probability* of $\mathcal{P}$ is an infinite sequence $\mathbf{p} = p_0 p_1 \ldots$, where $p_0 = s_0^\mathcal{P}$ and $p_k \to_\mathcal{P} p_{k+1}$ for all $k \geq 0$.

The acceptance condition for a trajectory of $\mathcal{P}$ is encoded in $\Omega_\mathcal{P}$, and is induced by the acceptance condition of $\mathcal{R}$. Formally, $\Omega_\mathcal{P}$ is a set of pairs $(\mathcal{F}_i^\mathcal{P}, \mathcal{B}_i^\mathcal{P})$, where $\mathcal{F}_i^\mathcal{P} = \{e \in \Delta_\mathcal{P} \mid \pi^{\Omega_\mathcal{R}}(e, \mathcal{F}_i) > 0\}$, $\mathcal{B}_i^\mathcal{P} = \{e \in \Delta_\mathcal{P} \mid \pi^{\Omega_\mathcal{R}}(e, \mathcal{B}_i) > 0\}$, and $(\mathcal{F}_i, \mathcal{B}_i) \in \Omega_\mathcal{R}$.

A trajectory of $\mathcal{P} = \mathcal{T} \times \mathcal{R}$ is said to be accepting if and only if there is a tuple $(\mathcal{F}_i^\mathcal{P}, \mathcal{B}_i^\mathcal{P}) \in \Omega_\mathcal{P}$ such that the trajectory intersects the sets $\mathcal{F}_i^\mathcal{P}$ and $\mathcal{B}_i^\mathcal{P}$ infinitely and finitely many times, respectively. It follows by construction that a trajectory $\mathbf{p} = (B_0, s_0)(B_1, s_1)\ldots$ of $\mathcal{P}$ is accepting if and only if the trajectory $\mathbf{s}^0_{0:T_0-1}\mathbf{s}^1_{0:T_1-1}\ldots$ is accepting in $\mathcal{R}$, where $\mathbf{s}^i_{0:T_i}$ is the random trajectory of $\mathcal{R}$ obtained by traversing the transition $e = (B_i, B_{i+1})$ using the controller $\mathcal{C}_\mathcal{T}(e)$ and $s_0^i = s_i$ for all $i \geq 0$. Note that $\mathbf{s}^i_{T_i} = \mathbf{s}^{i+1}_0$. As a result, a trajectory of $\mathcal{T}$ obtained from an accepting trajectory of $\mathcal{P}$ satisfies the given specification encoded by $\mathcal{R}$ with positive probability. We denote the projection of a trajectory $\mathbf{p} = (B_0, s_0)(B_1, s_1)\ldots$ onto $\mathcal{T}$ by $\mathbf{p}\lfloor_\mathcal{T} = B_0 B_1 \ldots$. A similar notation is used for projections of finite trajectories.

Remark 1. Note that the product MDP in Def. 5 is defined to be amenable to incremental operations with respect to the growth of the DTS, i.e., updating and checking for a solution of positive probability. This property is achieved by requiring the states of $\mathcal{P}$ to be reachable by transitions in $\Delta_\mathcal{P}$. The incremental update can be performed using a recursive procedure similar to the one described in (Vasile and Belta 2013).

Remark 2. The acceptance condition for $\mathcal{P}$ is defined by its transitions and not in the usual way in terms of its states, due to the stochastic nature of transitions between belief nodes in $\mathcal{T}$. We only record the initial and end DRA states of the DRA trajectories induced by the sample paths obtained using the local controllers.

4.5 Finding satisfying policies

The existence of a satisfying policy with positive probability can be checked efficiently on the product MDP $\mathcal{P}$ by maintaining end components EC[†] for induced subgraphs of $\mathcal{P}$ determined by the pairs in the acceptance condition $\Omega_\mathcal{P}$. For each pair $\mathcal{F}_i^\mathcal{P}, \mathcal{B}_i^\mathcal{P}$, let c_i denote the ECs associated with the graphs $G_i^\mathcal{P} = (S_\mathcal{P}, \Delta_\mathcal{P} \setminus \Gamma_i)$, where $\Gamma_i = \{(p, p') \in \Delta_\mathcal{P} \mid \pi^{\Omega_\mathcal{R}}(e, \mathcal{F}_i) = 0 \wedge \pi^{\Omega_\mathcal{R}}(e, \mathcal{B}_i) > 0, e = (p, p')\lfloor_\mathcal{T}\}$. Given c_i, checking for a satisfying trajectory in procedure $existsSatPolicy(\mathcal{P})$ becomes trivial. We test if there exists an EC that contains a transition (p, p') such that $\pi^{\Omega_\mathcal{R}}(e, \mathcal{F}_i) > 0$, where $e = (p, p')\lfloor_\mathcal{T}$. Note that we do not need to maintain $\Omega_\mathcal{P}$ explicitly, we only need to maintain the c_i. Efficient incremental algorithms to maintain these ECs were proposed in (Haeupler et al. 2012).

[†]An EC of an MDP is a sub-MDP such that there exists a policy such that each node in the EC can be reached from each other node in the EC with positive probability.

4.6 Dynamic program for Maximum Probability Policy

Given a GDTL-FIRM MDP, we can compute the optimal switching policy to maximize the probability that the given formula ϕ is satisfied. This is done by solving the following optimization problem

$$\operatorname*{arg\,max}_{m \in \mathcal{F}(S_{\mathcal{P}}, Act)} \bigwedge_{i \in \Omega_{\mathcal{R}}} \bigwedge_{j=1}^{\infty} Pr_m\left(\bigvee_{k=1}^{\infty} s_{j+k} \cap \mathcal{F}_i \wedge s_{j+k} \in S_{\mathcal{P}} \setminus B_i\right) \tag{17}$$

In other words, we find a policy that maximizes the probability of visiting the states in $\mathcal{F}_i$ infinitely often and avoiding $\mathcal{B}_i$. To find this policy, we first decompose $\mathcal{P}$ into a set of end components and find the accepting components. Since any sample path that satisfies ϕ must end in an accepting component, maximizing the probability of satisfying ϕ is equivalent to maximizing the probability of reaching such a component. The optimal policy is thus given by the relationship

$$\begin{aligned} J^{\infty}(s) &= \begin{cases} 1, & s \in c_i \\ \max\limits_{a \in Act(s)} \sum_{s'} \delta(s,a,s') J^{\infty}(s') & \text{else} \end{cases} \\ m(s) &= \operatorname*{arg\,max}_{a \in Act(s)} \sum_{s'} \delta(s,a,s') J^{\infty}(s') \end{aligned} \tag{18}$$

This can be solved by a variety of methods, including approximate value iteration and linear programming (Bertsekas 2012).

Depending on the probability of satisfying ϕ, the user may not wish to accept the solution. Alg 1 allows the user to specify a minimum satisfaction threshold, ϵ. If the probability of satisfaction after solving (18) is less than ϵ, or if no satisfying policy exists, the algorithm returns no solution.

4.7 Complexity

The overall complexity of maintaining the ECs used for checking for satisfying runs in $\mathcal{P}$ is $O(|\Omega_{\mathcal{R}}|\,|S_{\mathcal{P}}|^{\frac{3}{2}})$. The complexity bound is obtained using the algorithm described in (Haeupler et al. 2012) and is better by a polynomial factor $|S_{\mathcal{P}}|^{\frac{1}{2}}$ than computing the ECs at each step using a linear algorithm. Thus, checking for the existence of a satisfying run of positive probability can be done in $O(|\Omega_{\mathcal{R}}|)$ time. The dynamic programming algorithm is polynomial in $|S_{\mathcal{P}}|$ (Papadimitriou and Tsitsiklis 1987).

4.8 Analysis

In this section we show that the proposed algorithm is probabilistic complete under uncertainty (PCUU) in the sense from (Agha-mohammadi et al. 2014). In (Agha-mohammadi et al. 2014), a strong version of PCUU is defined which is intractable to achieve, because it requires searching over the entire space of admissible policies. Instead, similar to (Agha-mohammadi et al. 2014), we investigate a weaker version of PCUU that restricts the class of policies considered. Thus, the completeness guarantee is with respect to the existence of satisfying policies parameterized by the random state space graph underlying the TS $\mathcal{T}$.

Definition 6. (PCUU). Let $p_{\min} \in [0,1]$ be a probability bound, and ϕ a GDTL formula. Assume there exists a control policy $\mu^*(\cdot;\mathcal{T}^*, b_0)$ such that $Pr[\mathbf{b} \models \phi; \mu^*] > p_{\min}$. A sampling-based algorithm is said to be probabilistic complete under uncertainty (PCUU) if $\lim_{N\to\infty} Pr[\mathbf{b} \models \phi; \mu(\cdot;\mathcal{T}, B_0)] > p_{\min}$, where $\mu(\cdot;\mathcal{T}, b_0)$ is the maximum probability policy for $\mathcal{P} = \mathcal{T} \times \mathcal{R}$ starting from belief state b_0, and N is the number of belief nodes in $\mathcal{T}$.

Note that in the above definition we made the policy parametrization explicit.

Assumption 1. Assume that for all predicates $f \in F_\phi$ of a GDTL formula ϕ, the associated belief space regions $\mathcal{G}_f$ are open sets.

The following assumption and proposition are adapted from (Agha-mohammadi et al. 2014), and are given without proof.

Assumption 2. Assume that:

1. there is some time step N such that a local controller stops ec_e with a positive probability, i.e., $\exists N < \infty$ such that $\mathbb{P}_N(B_j \mid b, ec_e) > 0$ for all b;
2. local controllers $ec_e(\cdot; x_j)$ are Lipschitz continuous in their parameters x_j, the mean of the belief state that the controller converges to;
3. the belief transition pdf in (3) is Lipschitz continuous in the control u;
4. absorption regions (belief nodes) have bounded change in measure with respect to their controllers' parameters, i.e., given controller $ec(\cdot;x)$ and $ec'(\cdot;x')$ that define belief nodes B and B', there exists $r > 0$ and $c < \infty$ such that for $\|x - x'\| < r$ we have $\mathbb{P}_1(B \ominus B' \mid b, ec) < c\,\|x - x'\|$,

where $e = (B_i, B_j)$ is an edge, $\mathbb{P}_N(B \mid b, ec)$ is the transition probability from belief b into the set B under controller ec in at most $N \geq 1$ steps, $\ominus$ is the symmetric difference: $B \ominus B' = (B \setminus B') \cup (B' \setminus B)$.

The assumption captures the fact that local controllers should be able to drive the system to their goal regions in belief space. Moreover, local controllers and robot dynamics must satisfy regularity conditions that ensure incremental improvement of policies, i.e., continuity as captures by the following proposition.

Proposition 1. (Continuity of absorption probabilities). *Given Assumption 2 the absorption probability $\mathbb{P}(B_j \mid b, ec_e)$ is Lipschitz continuous in x_j for all $e = (B_i, B_j)$ and b, where (x_j, P^∞) is the center of B_j, and $\mathbb{P}(B \mid b, ec)$ is the transition probability of b into B under controller ec in any number of steps.*

Proposition 1, adapted from Proposition 1 from (Agha-mohammadi et al. 2014), establishes the continuity of absorption probabilities with respect to the randomly generated states used to compute $\mathcal{T}$. This is an important claim used to prove the probabilistic completeness under uncertainty of Algorithm 1, i.e., Theorem 1.

Theorem 1. *Algorithm 1 is PCUU.*

Proof. The proof of the theorem follows a similar argument to the one used in (Agha-mohammadi et al. 2014, Theorem 1).

Assume there exists a satisfying policy $\mu^*(\cdot; \mathcal{T}^*, b_0)$. Due to Assumption 1, we can define hyper-balls in the belief space $\mathcal{G}$ enclosing each belief node in $\mathcal{T}^*$ such that hyper-balls are entirely contained within the same predicate regions $\mathcal{G}_f$ as the associated belief nodes from $\mathcal{T}^*$. The hyper-balls are full-dimensional, and therefore have non-zero measure.

The algorithm only samples state estimates, while the stationary covariance matrices are computed based on the associated controllers. We are guaranteed that we can generate belief nodes within the hyper-balls due to Assumption 1 and by invoking Proposition 1. Note that the assumptions of the proposition are met by the Gaussian noise LTI setup considered in this paper.

Thus, as the number of samples grows unbounded, it follows that eventually the generated transition system $\mathcal{T}$ will contain belief nodes within each hyper-ball. This implies that there exists a control policy μ on $\mathcal{P} = \mathcal{T} \times \mathcal{R}$ such that $Pr[\mathbf{b} \models \phi; \mu] > p_{\min}$. Again, the last inequality follows from Proposition 1. □

5 Robot Tracking and Localization

This section explains our localization framework—the final piece of our solution tool chain—consisting of an aerial vehicle tracking the ground vehicle and estimating its location in reference to the map that was built in Sec. 3.2.

The ground robot executes its mission in the environment by traversing the transition system generated in the control synthesis phase while employing an Extended Kalman Filter (EKF) to estimate its position with measurements provided by the dedicated aerial vehicle. A localization marker on the ground robot includes two distinctly colored patches that aid in estimating its planar position and orientation in the environment frame. During localization, the quadrotor first localizes the centroid of each patch in the quadrotor's image frame as two image features, $(\mathbf{p}_1^q, \mathbf{p}_2^q)$. The quadrotor simultaneously calculates the rectified homography between the quadrotor's image frame (q) and the mosaic map image frame (m), i.e., $\mathbf{H}_{qm}^r$, to estimate the relative pose between the quadrotor and the map. The quadrotor projects the robot's pose in the image frame $(\mathbf{p}_1^q, \mathbf{p}_2^q)$ to the map frame $(\mathbf{p}_1^m, \mathbf{p}_2^m)$ using $\mathbf{H}_{qm}^r$. The quadrotor finally computes the ground robot's final pose in the environment frame (e), given by (x, y, θ), by linearly interpolating $(\mathbf{p}_1^m, \mathbf{p}_2^m)$ with the dimensions of the map image—in pixels—and the known dimensions of the environment—measured in meters. The centroid of the projected features yields the position, (x, y), while the orientation, θ, is calculated using the line that connects the two projected features.

Meanwhile, a 2D kinematic position-based visual servoing (PBVS) controller maneuvers the aerial robot to track the ground robot while simultaneously keeping sufficient overlap with the mosaic map image for an accurate homography estimation. Recall that the field-of-view of the individual cameras is not sufficient to view the entire environment, hence the requirement for the composite map image. Homography-based control drives the quadrotor into a desired position above the environment that is defined by the estimated position of the ground robot, (x, y). The quadrotor's position is further constrained to a rectangle, $R = [x_{min}, x_{max}] \times [y_{min}, y_{max}]$, where the boundaries of R affect the amount of desired overlap with the mosaic image. For example, setting the boundaries equal to the dimensions of the environment will drive the quadrotor directly over the ground robot, thus degrading the homography estimate when hovering near the environment's edges. Conversely, setting the boundaries equal to zero would keep the quadrotor coincident with the mosaic image frame and will lose coverage when the ground robot is near the edge of the environment. The ideal boundary values for a downward facing camera allows the camera to move just far enough to see the entire environment, i.e.,

$$\begin{bmatrix} x_{min} \\ y_{min} \end{bmatrix} = - \begin{bmatrix} x_{max} \\ y_{max} \end{bmatrix} = \begin{bmatrix} \frac{w_e - {}^e w_q}{2} \\ \frac{h_e - {}^e h_q}{2} \end{bmatrix}, \tag{19}$$

where (w_e, h_e) are the width and height of the environment in meters, $({}^e w_q, {}^e h_q)$ are the dimensions of the quadrotor's image frame, (w_q, h_q), after being projected into the environment frame. This projection is computed as,

$$\begin{bmatrix} {}^e w_q \\ {}^e h_q \\ A \end{bmatrix} = A\mathbf{K}^{-1} \begin{bmatrix} w_q \\ h_q \\ 1 \end{bmatrix}, \tag{20}$$

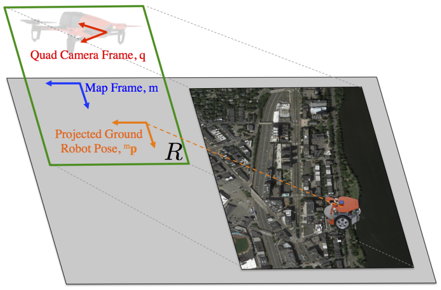


Figure 4. Coordinate frame definitions for the PBVS controller from equation (21) include the: environment frame, mosaic map image frame, quadrotor image frame, mosaic map frame center, and quadrotor frame. The quadrotor estimates the ground robot's pose (x, y, θ) by transforming the pose in the quadrotor image frame to the environment frame. The quadrotor manuevers within R based on the ground robots's pose in the environment frame. The quadrotor local frame and mosaic map frame center are defined with the same orientation as the environment frame.

given the camera's altitude, A, and camera calibration matrix, $\mathbf{K}$.

Finally, we introduce an optional offset, $\mathbf{x}_{offset}$, that measures the center of mosaic map image's virtual position in space with respect to the quadrotor's frame.

The final controller is similar to the homography-based formation controller in Section 3.2. In fact, the yaw controller of equation (8) and the altitude controller of equation (10) remain the same with a desired relative pose equal to zero. The planar control vector is calculated as the following,

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = K_v \left(\begin{bmatrix} [\mathbf{H}^r_{qm}]_{13} \\ [\mathbf{H}^r_{qm}]_{23} \end{bmatrix} - \begin{bmatrix} linint(x, (0, w_e), (x_{min}, x_{max})) - x_{offset} \\ linint(y, (0, h_e), (y_{min}, y_{max})) - y_{offset} \end{bmatrix} \right), \tag{21}$$

where $linint(\cdot)$ is the linear interpolation function that transforms the ground robot's environmental position into the quadrotor's desired position within R.

6 Experiments

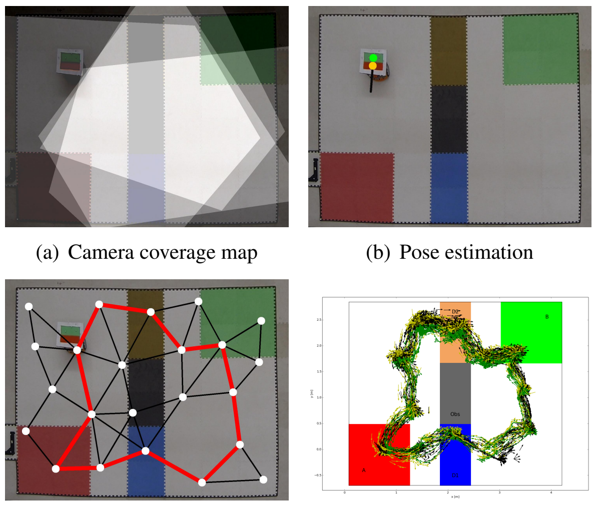
In this section, we present experimental results. First, we provide an overview of our experimental setup and hardware in Sec. 6.1. Next, we present the results of our case study using a ground robot and a fixed camera network in Sec. 6.2, followed by results from our entire end-to-end framework (Sec. 6.3), including results on mapping, control policy synthesis, and execution of the mission including localization in Secs. 6.3.1, 6.3.2, & 6.3.3, respectively.

6.1 Experimental Setup

We perform experiments in the Boston University Robotics Laboratory. We utilize an Optitrack motion capture system [‡] for obtaining ground truth measurements. The ground robot is a two-wheeled DrRobot X80Pro [§] with no onboard sensing. We fit the ground robot with an identifying marker composed of two uniquely colored patches in the YUV color space for planar position and orientation localization (see Fig. 5(b)). Parrot Bebop quadrotors[‖] are the aerial vehicles used for map building, and later, tracking. The Bebop is an off-the-shelf quadrotor platform with a suite of sensors that include an Inertial Measurement Unit (IMU), a downward-facing pinhole camera for optical flow stabilization, an ultrasonic sensor for altitude measurements, and a $180°$ wide-angle 14 megapixel forward-facing camera. The large forward-facing camera produces a 640×360 pixel stabilized video feed that can be 'steered' within the field-of-view of the wide-angle lens to produce a *virtual camera* video feed. We position the virtual camera at the maximum angle of θ_{bebop} measured about the y-axis of the quadrotor (see Fig. 6(a)), where $\theta_{bebop} \approx 50°$, and rectify the image for this angle. The ideal rectangle size for our camera at the desired experiment altitude of 1.8 meters is approximately 0.85×1.45 meters. Unfortunately, our camera is not downward-facing, therefore we expand R to 0.85×2.0 meters to ensure proper coverage. We use an offset 0.75 meters in the positive x-direction of the local quadrotor frame (see Fig. 4) to account for the forward-facing camera.

The algorithms in this paper were implemented in Python2.7 using *LOMAP* (Ulusoy et al. 2013) and *networkx* (Hagberg et al. 2008) libraries. The *ltl2star* tool (Klein and Baier 2006) was used to convert the LTL specification into a Rabin automaton. The Robot Operating System (ROS) (Quigley et al. 2009) handles all communication on a local area network via Wi-Fi. We control the quadrotors from a base station computer running the ROS *Bebop Autonomy* package (Monajjemi 2015) which incorporates Parrot's open-source SDK. The computer also acquires and processes image frames from the quadrotors' real-time video stream via the OpenCV libraries (Bradski 2000). Independent ROS nodes handle the individual quadrotors for the formation flight, demonstrating the distributed control. Independent ROS nodes also handle the quadrotor and ground robot control during the tracking phase. In this phase, separate quadrotor nodes handle the image processing for robot localization, pose estimation via homography, and the control. The ground robot node executes the local control and EKF estimation of the ground robot given its pose estimate and nonlinear dynamics. All vision computations are performed

[‡]Natural Point Optitrack: https://www.optitrack.com
[§]DrRobot X80Pro: http://www.drrobot.com/products_item.asp?itemNumber=x80pro
[‖]Parrot Bebop: http://www.parrot.com/products/bebop-drone/



(a) Camera coverage map (b) Pose estimation

(c) Transition system (d) Case study results

Figure 5. Fig. (a) shows the coverage of the cameras. Fig. (b) shows the pose of the robot computed from the images taken by the 4 cameras. Fig. (c) shows the transition system computed by Alg. 1. Fig. (d) shows the trajectory of the robot over 10 surveillance cycles. At each time step, the pose of the robot is marked by an arrow. The true trajectory of the robot is shown in green. The trajectory obtained from the camera network is shown in yellow, while the trajectory estimated by the Kalman filter is shown in black.

on an Ubuntu 14.04 machine with an Intel Core i7 CPU at 2.4 GHz and 8GB RAM.

6.2 Case Study

In this section, we present results from our running example (Example 2) to illustrate our algorithm. It is a simplified example that illustrates how the algorithm performs, and the type of specifications for which it is useful. The environment and specification are the same as those in Example 2. Localization is performed with a network of fixed IP cameras, whose coverage is shown in Fig. 5(a). The network was implemented using four TRENDnet Internet Protocol (IP) cameras with known pose with respect to the global coordinate frame of the experimental space. Each 640×400 RGB image is acquired and segmented, yielding multiple pixel locations that correspond to a known pattern on the robot.

A switched feedback policy was computed for the ground robot described by (14) operating in the environment shown in Fig. 3 with mission specification (6) using Alg. 1. The overall computation time to generate the policy was 32.739 seconds and generated a transition system (Fig. 5(c)) and product MDP of sizes (23, 90) and (144, 538), respectively. The Rabin automaton obtained from the GDTL formula has 7 states and 23 transitions operating over a set of atomic propositions of size 8. The most computationally intensive operation in Alg. 1 is the computation of the transition and intersection probabilities. To speed up the execution, we generated trajectories for each transition of the TS and reused them whenever Alg. 2 is called for a transition of the product MDP. The mean execution time for the probability computation was 0.389 seconds for each transition of $\mathcal{T}$.

We executed the computed policy on the ground vehicle over 9 experimental trials for a total of 24 surveillance cycles. The specification was met in all of surveillance cycles. A trajectory of the ground robot over 10 surveillance cycles (continuous operation) is shown in Fig. 5(d).

6.3 Experimental Setup

Now, we validate all three phases of our proposed framework by executing a complete mission experiment with a heterogeneous team of autonomous robots. The phases are completed in the order specified in Sections 3-5 due to the dependence on the results from previous phases. We first detail our map building results with a mosaic map that is generated using the homography-based formation control and two quadrotors with cameras that do not have access to GPS. GDTL-FIRM synthesizes the control policy for a ground robot with nonlinear *unicycle* dynamics in the environment for a GDTL specification over belief states associated with the measurement of the robot's position. Finally, a quadrotor successfully tracks and localizes the ground robot while it completes the previously defined mission.

We use a map of Boston University's campus, located in Boston, MA, USA, that includes parts of Charles River, Massachusetts Turnpike, Fenway Stadium, and BU Central campus. We utilize the real landmarks in this map to formulate our specification. This map is chosen because it has sufficient detail and texture to allow for adequate feature matching (e.g., white buildings at the bottom of the map) as well other minimal feature regions (e.g., the Charles River). The physical map is printed on a 12×16 ft^2 vinyl banner. Again, the Optitrack motion capture system provides ground truth measurements. The experimental setup contains actuation noise for both robots, slip due to the vinyl mat, and estimation noise. This noise allows us to demonstrate the viability of our framework. For real-world deployment, there may be a need for better actuators or quadrotor control to account for noise due to factors such as uneven terrain and wind.

6.3.1 Formation Control and Map Generation We utilize a team of two quadrotors to reach a desired formation where, $y^*_{1,2} = -y^*_{2,1} = 1.2$ m, and all other desired relative poses are set to zero (see Fig. 6(a)). This formation was carefully chosen because it ensures the pair of aerial cameras have enough overlap for accurate relative pose estimation while guaranteeing a complete view of the environment. All quadrotors are flown to a desired height of 1.8 meters. The quadrotors reach the desired formation (Fig. 6(c)) from the initial conditions (Fig. 6(b)) in approximately 15 seconds. From this point, the user has the ability to control one vehicle in the formation to fine tune the result of the online mosaic map, which is displayed at approximately 2.5Hz. In this experiment, the operator maneuvers quadrotor 1 until the left edge of the map is completely visible and then releases it to autonomous control again. Meanwhile, the formation control law in Section 3.2 controls quadrotor 2. The onboard images at the final desired formation (Figs. 6(d)- 6(e)) were used to generate the final mosaic map image shown in Fig. 6(f).

6.3.2 Control Policy Synthesis The specification for the ground robot is encoded with GDTL over the regions shown in Fig. 7 and is given as the following: “Always avoid all obstacles, i.e., Charles river and Massachusetts Turnpike. Always eventually visit Kenmore Square, Marsh Plaza, Audubon Circle, and Fenway Stadium. From Kenmore Square or Marsh Plaza, Bridge2 (St Mary’s St) can not be used to visit Audubon Circle or Fenway Stadium. From Audubon Circle or Fenway Stadium, Bridge1 (Beacon Ave or Brookline Ave) can not be used to visit Kenmore Square or Marsh Plaza. Always keep uncertainty about the robot’s pose below 0.9 m^2, and on bridges, the uncertainty must be below 0.6 m^2, where uncertainty is measured as the trace of the estimation pose covariance matrix.” Fig. 8(a) shows the resulting transition system and control policy, computed by the algorithm from Alg. 1. The transition system has 35 nodes and 226 edges while the product automaton has 316 nodes and 3274 edges. The algorithm executed in approximately 62.24 seconds.

6.3.3 Pose Estimation and Mission Execution The ground robot executes the mission using the previously computed control policy and quadrotor for localization. Initially, the quadrotor takes off from a position where the camera’s field of view is facing towards the ground robot. The homography-based localization and quadrotor control (Section 5) begin once the ground robot’s marker has been detected. The ground robot localization estimates update at approximately 3.5Hz. We show an example of the robot tracking and pose estimation for three time steps in Fig. 9. It is clear that the control method tracks the ground robot during its route with enough image resolution to detect the robot’s patches and also maintains the required overlap with the mosaic map image.

Fig. 9 also illustrates the final pose estimation in the mosaic map frame. It is important to note that the ground robot sits 0.2 meters above the map, therefore projecting the image features of the ground robot’s marker directly into the map frame would add significant error to the final estimation. The image features are instead offset to the map plane before projecting the features to the mosaic map to satisfy the homography’s planar assumption. We determine this offset by measuring the pose estimation error at the extremes of the map and interpolating for the correction as a function of the estimated pose.

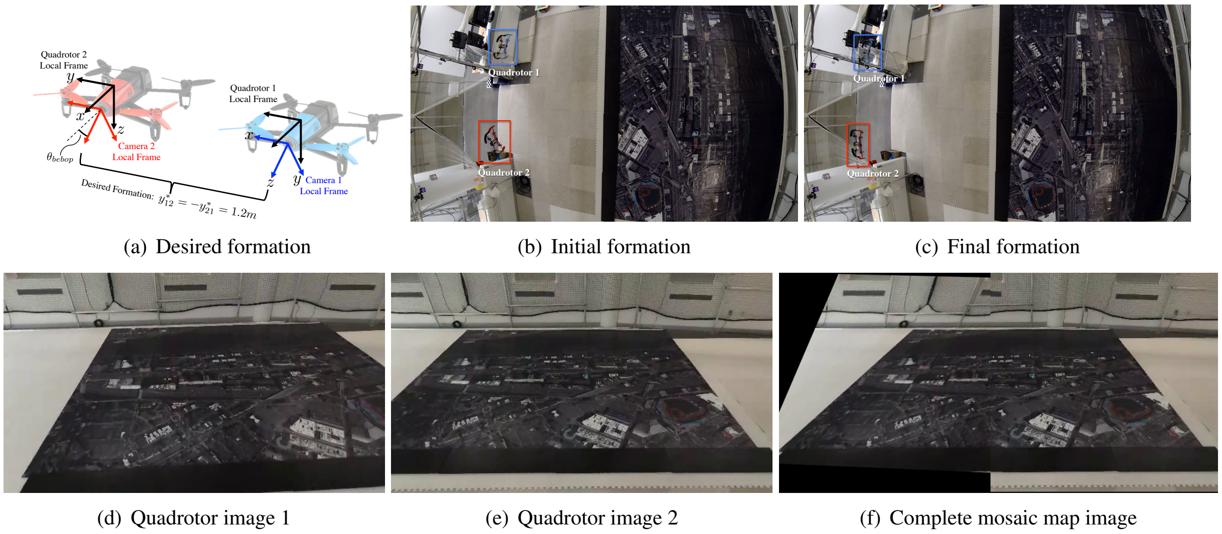
We ran the mission five times due to the limitations of the quadrotor battery, yielding ten complete laps of the environment and four partial laps, all of which satisfied the GDTL specification. Each time the quadrotor battery was replaced, the ground robot position estimate is re-initialized, resulting in the four covariance spikes over the ten laps (Fig. 8(d)). We show an example run of 2.5 laps in Fig. 8(b) that displays the ground robot’s ground truth pose, estimated pose, measured pose, and uncertainty. We check for satisfaction by inspecting the ground truth of all experimental runs to ensure the robot has reached each region appropriately while avoiding obstacles (Fig. 8(c)). Moreover, the covariance of the robot’s estimate for all experimental runs is safely below the minimum 0.6 requirement, thus satisfying the specification (Fig. 8(d)). Note that the the covariance quickly converges to a small value, but oscillates due to the nonlinear approximated Bayesian filter. These oscillations are visible in the detail in Fig. 8(e).

7 Conclusion

In this paper, we presented an end-to-end framework for mapping an unknown GPS-denied environment, synthesizing a control policy for a noisy ground robot, and executing that control policy using a quadrotor to localize the ground robot.

Synthesis was performed using a sampling-based algorithm that generates feedback policies for stochastic systems with temporal and uncertainty constraints. The desired behavior of the system is specified using *Gaussian Distribution Temporal Logic* such that the generated policy satisfies the task specification with maximum probability. The proposed algorithm generates a transition system in the belief space of the system. A key step towards the scalability of the automata-based methods employed in the solution was breaking the *curse of history* for POMDPs. Local feedback controllers that drive the system within belief sets were employed to achieve history independence for paths in the transition system. Also contributing to the scalability of our solution is a construction procedure for an annotated product Markov Decision Process called GDTL-FIRM, where each transition is associated with a “failure probability”. GDTL-FIRM captures both satisfaction and the stochastic behavior of the system. Switching feedback policies were computed over the product MDP.

We experimentally tested the multi-part framework, including building a map with two quadrotors, using the map to synthesize a control policy for a ground robot,



(a) Desired formation (b) Initial formation (c) Final formation
(d) Quadrotor image 1 (e) Quadrotor image 2 (f) Complete mosaic map image

Figure 6. Final mosaic map result using the homography-based formation control method.

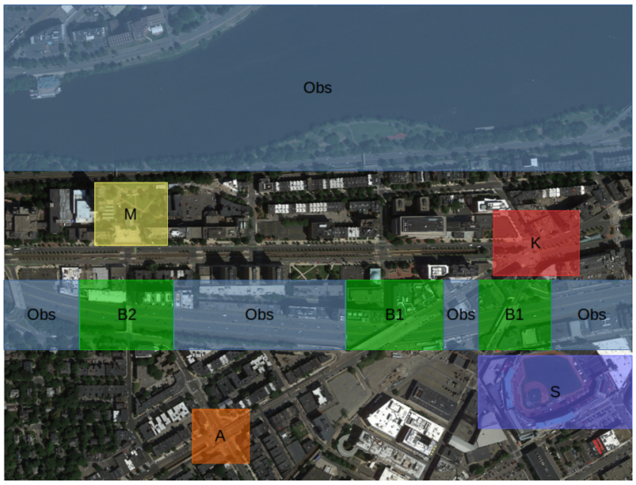
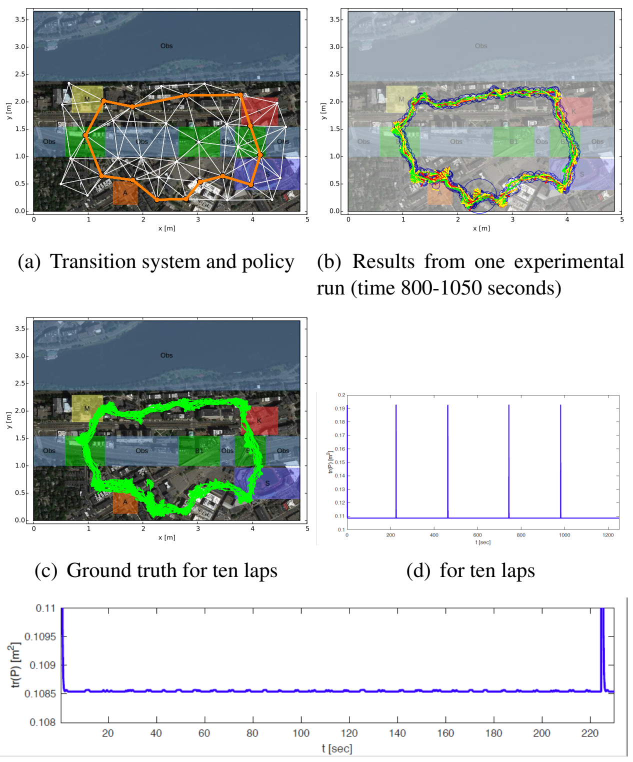


Figure 7. Map with labels for mission specification.

then executing the policy on the ground robot using an aerial robot for localization. The experiments showed that properties specifying the temporal and stochastic behavior of systems can be expressed using GDTL and our algorithm is able to compute control policies that satisfy the specification with a given probability.

There is much potential for future work. First, as mentioned above, the quadrotor could perform noise-aware localization by adjusting its position and altitude to provide better localization as needed. Such active localization complicates the control problem, but could aid in the synthesis of better policies. For example, if uncertainty is too large for the ground robot to safely accomplish its task, the quadrotor could maneuver to decrease the ground robot's uncertainty and lead to higher probability of satisfaction. Such an approach would complicate the synthesis problem by expanding the dimensionality of state and action spaces under consideration, but could provide substantially stronger real-world capabilities.

Another direction for future work is reactive sampling and planning. This work considers an environment that is not changing over time, and therefore the motion plan never changes. In many real world applications this is not the case, and the robot must react in real-time to moving elements in the environment. Dealing with a dynamic environment would require the ability to re-plan online. Therefore, our framework would require close to real-time sampling and synthesis capability. The incremental approach provided by sampling is promising in this regard. Online synthesis could also be useful in an information gathering setting, in which the specification includes uncertainty about a sensing goal for the ground robot. Such a specification might be to localize a chemical spill, for example. Satisfying that specification would require closing the loop with the ground robot's sensing capabilities. Finally, modifying the vision framework to include more advanced techniques, such as structure from motion, could improve performance in areas that are less feature rich than our map, such as urban environments from a high altitude. In terms of other changes to the computer vision pipeline, we could also consider refining the mapping or target tracking components with a more modern Convolutional Neural Network approach. For example, there are new Bayesian filtering methods that consider the entire image as the measurement (rather than just 2D point features) and learn a useful latent measurement representation for a Kalman filter via a deep neural network (Haarnoja et al. 2016). Such approaches would require modifying our estimation



(a) Transition system and policy (b) Results from one experimental run (time 800-1050 seconds)

(c) Ground truth for ten laps (d) for ten laps

(e) Covariance for the first 200 seconds

Figure 8. FIRM-GDTL results plotted over the ground truth environment image. Fig. 8(a) shows the transition system in white and the policy in orange. Fig. 8(b) shows the ground truth in green, the measurement in yellow, the estimated pose in red, and the covariance ellipses in blue. Fig. 8(c) shows the ground truth in green for all runs. Fig. 8(d) shows the covariance for all runs. The spikes in covariance indicate the beginning of a new run after a quadrotor battery had been replaced. We initialize the covariance to an arbitrarily large value at time step 0 that drastically decreases with the first pose measurement from the quadrotor at time step 1. Fig. 8(e) shows oscillations due to the nonlinear approximated Bayesian filter.

pipeline to accomodate and take full advantage of the types of information provided by such a filter.

References

Agha-mohammadi, A., Chakravorty, S. and Amato, N. (2014), 'Firm: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements', *The International Journal of Robotics Research* **33**(2), 268–304.

Ayala, A. M., Andersson, S. B. and Belta, C. (2014), 'Formal Synthesis of Control Policies for Continuous Time Markov Processes From Time-Bounded Temporal Logic Specifications', *IEEE Transactions on Automatic Control* **59**(9), 2568–2573.

Bachrach, A., Prentice, S., He, R., Henry, P., Huang, A., Krainin, M., Maturana, D., Fox, D. and Roy, N. (2012), 'Estimation, planning, and mapping for autonomous flight using an RGB-D camera in GPS-denied environments', *The International Journal of Robotics Research* **31**(11), 1320–1343.

Benhimane, S. and Malis, E. (2006), Homography-based 2d visual servoing, *in* 'Proceedings of the 2006 International Conference on Robotics and Automation (ICRA)', IEEE, pp. 2397–2402.

Bertsekas, D. (2012), *Dynamic Programming and Optimal Control*, 3rd,4th edn, Athena Scientific.

Bradski, G. (2000), 'The opencv library', *Doctor Dobbs Journal* **25**(11), 120–126.

Bry, A. and Roy, N. (2011), Rapidly-exploring random belief trees for motion planning under uncertainty, *in* 'Proceedings of the 2011 International Conference on Robotics and Automation (ICRA)', IEEE, pp. 723–730.

Burns, B. and Brock, O. (2007), Sampling-based motion planning with sensing uncertainty, *in* 'Proceedings of the 2007 International Conference on Robotics and Automation', IEEE, pp. 3313–3318.

Campbell, M. E. and Whitacre, W. W. (2007), 'Cooperative tracking using vision measurements on seascan uavs', *IEEE Transactions on Control Systems Technology* **15**(4), 613–626.

Cognetti, M., Salaris, P. and Giordano, P. R. (2018), Optimal active sensing with process and measurement noise, *in* '2018 IEEE International Conference on Robotics and Automation (ICRA)', pp. 2118–2125.

Cristofalo, E., Leahy, K., Vasile, C.-I., Montijano, E., Schwager, M. and Belta, C. (2016), Vision-based mobile sensing for gps-deprived control with temporal logic specifications, *in* 'Proceedings of the International Symposium on Experimental Robotics (ISER 16)'.

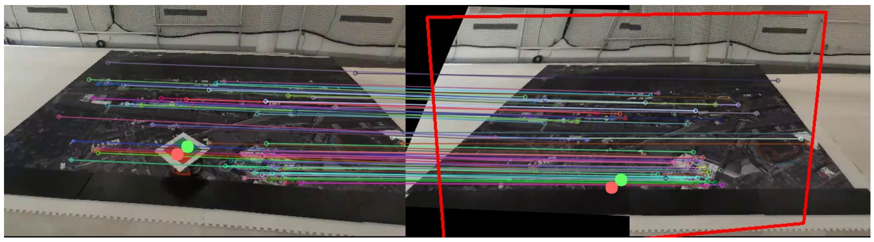
Davison, A. J., Reid, I. D., Molton, N. D. and Stasse, O. (2007), 'Monoslam: Real-time single camera slam', *IEEE Transactions on Pattern Analysis & Machine Intelligence* **29**(6), 1052–1067.

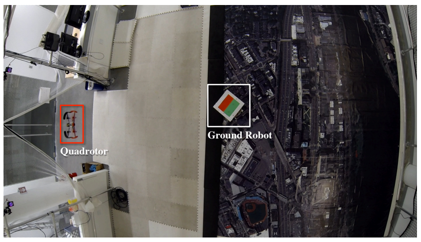
Forster, C., Pizzoli, M. and Scaramuzza, D. (2013), Air-ground localization and map augmentation using monocular dense reconstruction, *in* 'Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 3971–3978.

Grocholsky, B., Keller, J., Kumar, V. and Pappas, G. (2006), 'Cooperative air and ground surveillance', *Robotics & Automation Magazine* **13**(3), 16–25.

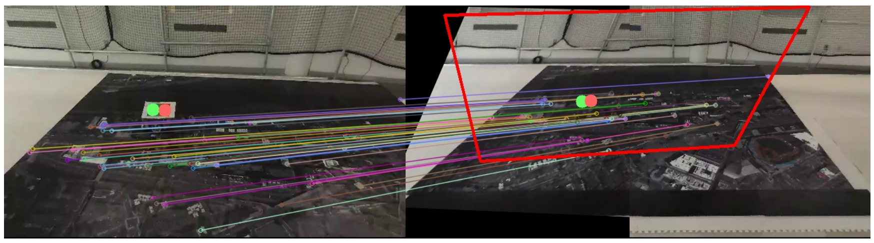
Haarnoja, T., Ajay, A., Levine, S. and Abbeel, P. (2016), Backprop kf: Learning discriminative deterministic state estimators, *in* 'Advances in Neural Information Processing Systems', pp. 4376–4384.

Haeupler, B., Kavitha, T., Mathew, R., Sen, S. and Tarjan, R. (2012), 'Incremental Cycle Detection, Topological Ordering, and Strong Component Maintenance', *ACM Trans.*

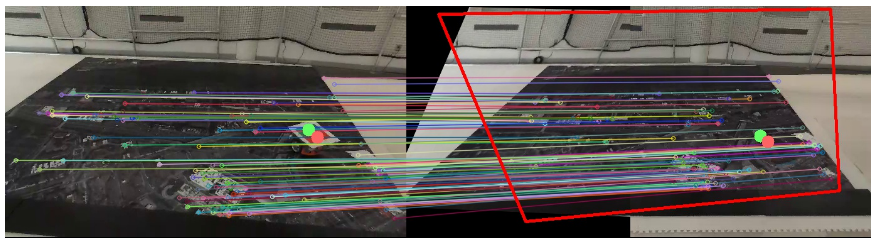

(a) Pose estimate at time = 814 seconds

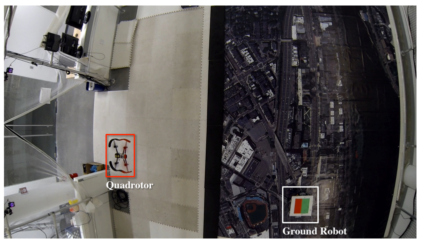


(b) Top view


(c) Pose estimate at time = 892 seconds

(d) Top view


(e) Pose estimate at time = 964 seconds



(f) Top view

Figure 9. Pose estimation results of live tracking and localization are shown in Figs. 5(a,c,e) with onboard images (left) and the mosaic map image (right). The corresponding top views of the experiment is shown in Figs. 5(b,d,f), respectively. The image matches and pose estimations are drawn for visualization purposes.

Algorithms **8**(1), 1–33.

Hagberg, A., Schult, D. and Swart, P. (2008), Exploring network structure, dynamics, and function using NetworkX, *in* 'Proceedings of the 7th Python in Science Conference (SciPy2008)', Pasadena, CA USA, pp. 11–15.

Hauser, K. (2011), Randomized belief-space replanning in partially-observable continuous spaces, *in* 'Algorithmic Foundations of Robotics IX', Springer, pp. 193–209.

Hsieh, M. A., Cowley, A., Keller, J., Chaimowicz, L., Grocholsky, B., Kumar, V., Taylor, C., Endo, Y., Arkin, R., Jung, B. and Wolf, D. F. (2007), 'Adaptive teams of autonomous aerial and ground robots for situational awareness', *Journal of Field Robotics* **24**(11-12), 991–1014.

Jones, A., Schwager, M. and Belta, C. (2013), Distribution temporal logic: Combining correctness with quality of estimation, *in* 'Proceedings of the IEEE Conference on Decision and Control (CDC)', pp. 4719–4724.

Kaelbling, L., Littman, M. and Cassandra, A. (1998), 'Planning and acting in partially observable stochastic domains', *Artificial Intelligence* **101**(1–2), 99 – 134.

Kalman, R. E. (1960), 'A new approach to linear filtering and prediction problems', *Journal of basic Engineering* **82**.

Karaman, S. and Frazzoli, E. (2009), Sampling-based Motion Planning with Deterministic μ-Calculus Specifications, *in* 'Proceedings of the IEEE Conference on Decision and Control (CDC)', Shanghai, China.

Karaman, S. and Frazzoli, E. (2011), 'Sampling-based Algorithms for Optimal Motion Planning', *International Journal of Robotics Research* **30**(7), 846–894.

Karaman, S. and Frazzoli, E. (2012), Sampling-based Optimal Motion Planning with Deterministic μ-Calculus Specifications, *in* 'Proceedings of the American Control Conference (ACC)'.

Klein, J. and Baier, C. (2006), Experiments with deterministic ω-automata for formulas of linear temporal logic, *in* 'Implementation and Application of Automata', Springer, pp. 199–212.

Lahijanian, M., Maly, M. R., Fried, D., Kavraki, L. E., Kress-Gazit, H. and Vardi, M. Y. (2016), 'Iterative temporal planning in uncertain environments with partial satisfaction guarantees', *IEEE Transactions on Robotics* **32**(3), 538–599.

LaValle, S. M. (2006), *Planning Algorithms*, Cambridge University Press, Cambridge, U.K. Available at http://planning.cs.uiuc.edu/.

Leahy, K., Jones, A., Schwager, M. and Belta, C. (2015), Distributed information gathering policies under temporal logic constraints, *in* 'Proceedings of the IEEE Conference on Decision and Control (CDC)', pp. 6803–6808.

Lesser, K. and Oishi, M. (2015), Finite State Approximation for Verification of Partially Observable Stochastic Hybrid Systems, *in* 'Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control', ACM, New York, NY, USA, pp. 159–168.

Li, X. R. and Jilkov, V. P. (2001), Survey of maneuvering target tracking: Iii. measurement models, *in* 'Signal and Data Processing of Small Targets 2001', Vol. 4473, International Society for Optics and Photonics, pp. 423–447.

Ma, Y., Soatto, S., Kosecka, J. and Sastry, S. S. (2004), *An Invitation to 3D Vision: From Images to Geometric Models*, Springer.

Maly, M., Lahijanian, M., Kavraki, L., Kress-Gazit, H. and Vardi, M. (2013), Iterative temporal motion planning for hybrid systems in partially unknown environments, *in* '16th International Conference on Hybrid Systems: Computation and Control', ACM, pp. 353–362.

Monajjemi, M. (2015), 'Bebop autonomy', `https://github.com/AutonomyLab/bebop_autonomy`.

Montijano, E., Cristofalo, E., Zhou, D., Schwager, M. and Sagues, C. (2016), 'Vision-based distributed formation control without an external positioning system', *Transactions on Robotics* **32**(1), 339–351.

Mueggler, E., Faessler, M., Fontana, F. and Scaramuzza, D. (2014), Aerial-guided navigation of a ground robot among movable obstacles, *in* 'Safety, Security, and Rescue Robotics (SSRR), 2014 IEEE International Symposium on', IEEE, pp. 1–8.

Papadimitriou, C. and Tsitsiklis, J. (1987), 'The complexity of markov decision processes', *Mathematics of operations research* **12**(3), 441–450.

Patil, S., Kahn, G., Laskey, M., Schulman, J., Goldberg, K. and Abbeel, P. (2015), Scaling up Gaussian Belief Space Planning Through Covariance-Free Trajectory Optimization and Automatic Differentiation, *in* 'Algorithmic Foundations of Robotics XI', Springer, pp. 515–533.

Prentice, S. and Roy, N. (2009), 'The belief roadmap: Efficient planning in belief space by factoring the covariance', *The International Journal of Robotics Research* .

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R. and Ng, A. (2009), Ros: an open-source robot operating system, *in* 'ICRA workshop on open source software', Vol. 3, p. Page 5.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. and Berg, A. (2015), 'Imagenet large scale visual recognition challenge', *International Journal of Computer Vision* **115**(3), 211–252.

Simonyan, K. and Zisserman, A. (2014), 'Very deep convolutional networks for large-scale image recognition', *arXiv preprint arXiv:1409.1556* .

Svorenova, M., Cerna, I. and Belta, C. (2013), Optimal control of mporal logic constraints, *in* 'Proceedings of the IEEE Conference on Decision and Control (CDC)', pp. 3938–3943.

Thrun, S., Burgard, W. and Fox, D. (2005), *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*, The MIT Press.

Ulusoy, A., Smith, S. L., Ding, X. C., Belta, C. and Rus, D. (2013), 'Optimality and Robustness in Multi-Robot Path Planning with Temporal Logic Constraints', *International Journal of Robotics Research* **32**(8), 889–911.

van den Berg, J., Abbeel, P. and Goldberg, K. (2011), 'LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information', *The International Journal of Robotics Research* **30**(7), 895–913.

Vasile, C. and Belta, C. (2013), Sampling-Based Temporal Logic Path Planning, *in* 'Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', Tokyo.

Vasile, C. and Belta, C. (2014), Reactive Sampling-Based Temporal Logic Path Planning, *in* 'Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)', Hong Kong.

Vasile, C., Leahy, K., Cristofalo, E., Jones, A., Schwager, M. and Belta, C. (2016), Control in belief space with temporal logic specifications, *in* 'Proceedings of the 2016 Conference on Decision and Control (CDC)', IEEE.

Vaughan, R., Sukhatme, G., Mesa-Martinez, F. and Montgomery, J. (2000), Fly spy: Lightweight localization and target tracking for cooperating air and ground robots, *in* 'Distributed autonomous robotic systems 4', Springer, pp. 315–324.

Vitus, M. P. and Tomlin, C. J. (2011), Closed-loop belief space planning for linear, gaussian systems, *in* 'Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)', pp. 2152–2159.

Zamani, M., Esfahani, P. M., Majumdar, R., Abate, A. and Lygeros, J. (2014), 'Symbolic control of stochastic systems via approximately bisimilar finite abstractions', *IEEE Transactions on Automatic Control* **59**(12), 3135–3150.