The International Journal of Robotics Research
1–18
© The Author(s) 2014
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0278364914522312
ijr.sagepub.com

Control of noisy differential-drive vehicles from time-bounded temporal logic specifications

Igor Cizelj and Calin Belta

Abstract
We address the problem of controlling a noisy differential drive mobile robot such that the probability of satisfying a specification given as a bounded linear temporal logic formula over a set of properties at the regions in the environment is maximized. We assume that the vehicle can determine its precise initial position in a known map of the environment. However, motivated by practical limitations, we assume that the vehicle is equipped with noisy actuators and, during its motion in the environment, it can only measure the angular velocity of its wheels using limited accuracy incremental encoders. Assuming the duration of the motion is finite, we map the measurements to a Markov decision process (MDP). We use recent results in statistical model checking to obtain an MDP control policy that maximizes the probability of satisfaction. We translate this policy to a vehicle feedback control strategy and show that the probability that the vehicle satisfies the specification in the environment is bounded from below by the probability of satisfying the specification on the MDP. We illustrate our method with simulations and experimental results.

Keywords
Formal methods, reactive and sensor-based planning

1. Introduction

Robot motion planning and control has been widely studied in the last twenty years. In “classical” motion planning problems (LaValle, 2006), the specifications are usually restricted to simple primitives of the type “go from A to B and avoid obstacles”, where A and B are two regions of interest in some environment. Recently, temporal logics, such as linear temporal logic (LTL) and computational tree logic (CTL) (Baier and Katoen, 2008; Clarke et al., 1999) have become increasingly popular for specifying robotic tasks (see, for example, Loizou and Kyriakopoulos, 2004; Kress-Gazit et al., 2007; Karaman and Frazzoli, 2008; Kloetzer and Belta, 2008b; Fainekos et al., 2009; Wongpiromsarn et al., 2009; Bhatia et al., 2011). It has been shown that temporal logics can serve as rich languages capable of specifying complex motion missions such as “go to region A and avoid region B unless regions C or D are visited.”

In order to use existing model checking tools for motion planning (see Baier and Katoen, 2008), many of the above-mentioned works rely on the assumption that the motion of the vehicle in the environment can be modeled as a finite system (Clarke et al., 1999) that is either deterministic (applying an available action triggers a unique transition (see, for example, Ding et al., 2012)) or nondeterministic (applying an available action can enable multiple transitions, with no information on their likelihoods (see, for example, Kloetzer and Belta, 2008a)). Recent results show that, if sensor and actuator noise models can be obtained from empirical measurements or an accurate simulator, then the robot motion can be modeled as a Markov decision process (MDP), and probabilistic temporal logics, such as probabilistic CTL (PCTL) and probabilistic LTL (PLTL), can be used for motion planning and control (see Lahijanian et al., 2012).

However, robot dynamics are normally described by control systems with state and control variables evaluated over infinite domains. A widely used approach for temporal logic verification and control of such a system is through the construction of a finite abstraction (Tabuada and Pappas, 2006; Girard, 2007; Kloetzer and Belta, 2008b; Yordanov et al., 2012). Even though recent works discuss the construction of abstractions for stochastic systems (see, for example, D’Innocenzo et al., 2008; Julius and Pappas, 2009; Abate et al., 2011), the existing methods are either not applicable

Division of Systems Engineering at Boston University, MA, USA

Corresponding author:
Igor Cizelj, Division of Systems Engineering at Boston University, Boston, MA 02215, USA.
Email: icizelj@bu.edu

due to nonlinearities in robot dynamics or are computationally very expensive due to the partition-based abstractions of the state and control spaces.

In this paper, we consider a vehicle whose performance is measured by the completion of time constrained temporal logic tasks. In particular, we provide a conservative solution to the problem of controlling a stochastic differential drive mobile robot such that the probability of satisfying a specification given as a bounded linear temporal logic (BLTL) formula over a set of properties at the regions in the environment is maximized. We assume that the vehicle can determine its precise initial position in a known map of the environment. The actuator noise is modeled as a random variable with a continuous probability distribution supported on a bounded interval, where the distribution is obtained through experimental trials. Also, we assume that the vehicle is equipped with two limited accuracy incremental encoders, each measuring the angular velocity of one of the wheels, as the only means of measurement available. These assumptions are motivated by realistic robotic applications with communication constraints, e.g. in GPS-denied environments. For example, the robot can use GPS only from time to time to localize itself on a known map of the environment. In between GPS readings, the robot uses its (noisy) incremental encoders and maximizes the probability of satisfying the specification until a new GPS reading can be made.

Assuming the duration of the motion is finite, through discretization, we map the incremental encoder measurements to an MDP. By relating the MDP to the vehicle motion in the environment, the vehicle control problem becomes equivalent to the problem of finding a control policy for an MDP such that the probability of satisfying the BLTL formula is maximized. Due to the size of the MDP, finding the exact solution is prohibitively expensive. We trade-off correctness for scalability, and we use computationally efficient techniques based on sampling. Specifically, we use recent results in statistical model checking (SMC) for MDPs (Henriques et al., 2012) to obtain an MDP control policy and a bayesian interval estimation (BIE) algorithm (Zuliani et al., 2010) to estimate the probability of satisfying the specification. We show that the probability that the vehicle satisfies the specification in the original environment is bounded from below by the maximum probability of satisfying the specification on the MDP under the obtained control policy.

The main contribution of this work lies in bridging the gap between low level sensory inputs and high level temporal logic specifications. We develop a framework for the synthesis of a vehicle feedback control strategy from such specifications based on a realistic model of an incremental encoder. This paper extends our previous work (Cizelj and Belta, 2012) of controlling a stochastic version of a Dubins vehicle such that the probability of satisfying a temporal logic statement, given as a PCTL formula, over some environmental properties, is maximized. Specifically, the approach presented here allows for richer temporal logic specifications, where the vehicle performance is measured by the completion of time constrained temporal logic tasks. Additionally, in order to deal with the increase in the size of the problem we use computationally efficient techniques based on sampling. In Henriques et al., 2012, the authors use SMC for MDPs to solve a motion planning problem for a vehicle moving on a finite grid, assuming perfect localization,when the task is given as a BLTL formula. We adopt this approach to control a vehicle with continuous dynamics and allowing for uncertainty in its state. Some of the results in this paper were presented without proofs in Cizelj and Belta, 2013. In this paper, we include all the technical details and a complexity analysis of the overall approach. In addition, this paper contains a detailed description of our experimental validations and two more case studies.

This work is related to Lahijanian et al., 2012, where the problem of maximizing the probability of satisfying a PCTL formula over a set of regions was considered. In Lahijanian et al., 2012, the MDP model of the robot was constructed by partitioning the environment and performing extensive experiments and simulations. The approach proposed here avoids this expensive process. This paper is also closely related to "classical" dynamic programming (DP)-based approaches (Alterovitz et al., 2007). In these problems, the set of allowed specifications is restricted to reaching a given destination state, whereas our BLTL control framework allows for richer, temporal logic specifications and multiple destinations. In Shkolnik et al., 2009, the authors solve the problem of reaching a given destination while avoiding obstacles using the rapidly exploring random tree (RRT) algorithm that takes into account local reachability, as defined by differential constraints. In our approach, when relating the MDP to the vehicle motion in the environment, reachable sets are also considered, but we take into account reachability under uncertainty. Moreover, our approach produces a feedback control strategy and a lower bound on the probability of satisfaction, whereas the method presented in Shkolnik et al., 2009 only returns a collision-free trajectory. In addition, these methods differ from our work since they require precise state of the vehicle, at all times. One approach for planning under motion uncertainty (Melchior and R.Simmons, 2007), called the particle RRT algorithm, explicitly considers uncertainty in its domain. Each extension to the search tree is treated as a stochastic process and is simulated multiple times. Even though this method produces a collision-free trajectory together with the probability of following the path, it is restricted to a simple task of reaching a given destination state and requires precise state information. In Grady et al., 2013 the authors solve a motion planning problem for a car-like vehicle under uncertainty by posing the problem as a partially observable MDP (POMDP). In particular, the framework finds a policy that provides the optimal action given all past noisy sensor observations, while abstracting some of the motion constraints to reduce computation time. The same problem has been solved in Bai et al., 2013 where the authors present a simple algorithm for offline POMDP planning in the continuous state space that produces a POMDP policy, which can be executed efficiently online as a finite-state controller. However, these approaches are restricted to simple motion planning problems ("go from A to B while avoiding obstacles").

The remainder of the paper is organized as follows. In Section 2, we introduce the necessary notation and review some preliminaries. We formulate the problem and outline the approach in Section 3. In Sections 4–7 we explain the construction of the MDP and the relation between the MDP and the motion of the vehicle in the environment. The vehicle control strategy is obtained in Section 8. Case studies and experimental results illustrating our approach are presented in Section 9. We conclude with final remarks and directions for future work in Section 10.

2. Preliminaries

In this section we provide a short and informal introduction to MDP and BLTL. For details about MDPs the reader is referred to Baier and Katoen (2008) and for more information on BLTL to Jha et al. (2009) and Zuliani et al. (2010).

Definition 2.1 (MDP). *An MDP is a tuple* $M = (S, s_0, Act, A, P)$*, where* S *is a finite set of states;* $s_0 \in S$ *is the initial state;* Act *is a finite set of actions;* $A : S \rightarrow 2^{Act}$ *is a function specifying the enabled actions at a state* s*;* $P : S \times Act \times S \rightarrow [0,1]$ *is a transition probability function such that for all states* $s \in S$ *and actions* $a \in A(s)$*:* $\sum_{s' \in S} P(s,a,s') = 1$*, and for all actions* $a \notin A(s)$ *and* $s' \in S$*:* $P(s,a,s') = 0$.

A control policy for an MDP resolves nondeterminism in each state s by providing a distribution over the set of actions enabled in s.

Definition 2.2 (MDP control policy). *A control policy* μ *of an MDP* M *is a function* $\mu : S \times Act \rightarrow [0,1]$*, s.t.,* $\sum_{a \in A(s)} \mu(s,a) = 1$ *and* $\mu(s,a) > 0$ *only if* a *is enabled in* s*. A control policy for which either* $\mu(s,a) = 1$ *or* $\mu(s,a) = 0$ *for all pairs* $(s,a) \in S \times Act$ *is called deterministic.*

A control policy μ resolves all nondeterministic choices in an MDP. Given an initial state s_0, an action $a \in A(s_0)$ is applied with probability $\mu(s_0, a)$, and once the action is applied, the next state s' is obtained by sampling from the probability distribution P, i.e. the next state is s' with probability $P(s_0, a, s')$, and so on.

We employ BLTL to describe high level motion specifications. BLTL is a variant of LTL (Baier and Katoen, 2008), for which it has been shown that finite simulations of bounded duration are always sufficient for model checking (Zuliani et al., 2010). A detailed description of the syntax and semantics of BLTL is beyond the scope of this paper and can be found in Jha et al. (2009) and Zuliani et al. (2010). Roughly, formulas of BLTL are constructed by connecting properties from a set of proposition Π using Boolean operators ($\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction)), and temporal operators ($\mathbf{U}^{\leq t}$ (bounded until), $\mathbf{F}^{\leq t}$ (bounded finally), and $\mathbf{G}^{\leq t}$ (bounded globally), where $t \in \mathbb{R}^{\geq 0}$ is the time bound parameter and $\mathbb{R}^{\geq 0}$ denotes the set of nonnegative real numbers). The semantics of BLTL formulas are given over infinite traces $\sigma = (o_1, t_1)(o_2, t_2)\ldots$, $o_i \in 2^{\Pi}$, $t_i \in \mathbb{R}^{\geq 0}$, $i \geq 1$, where o_i is the set of satisfied propositions and t_i is the time spent satisfying o_i. However, in Zuliani et al. (2010) the authors show that model checking a trace on a BLTL formula is well-defined and can be based on only a finite prefix of the trace of bounded duration. A trace satisfies a BLTL formula ϕ if ϕ is true at the first position of the trace; $\mathbf{F}^{\leq t}\phi_1$ means that ϕ_1 will be true within t time units; $\mathbf{G}^{\leq t}\phi_1$ means that ϕ_1 will remain true for the next t time units; and $\phi_1 \mathbf{U}^{\leq t} \phi_2$ means that ϕ_2 will be true within the next t time units and ϕ_1 remains true until then. More expressivity can be achieved by combining the above temporal and Boolean operators.

3. Problem formulation and approach

3.1. Problem formulation

Motion model: A differential drive mobile robot (LaValle, 2006) is a vehicle having two main wheels, each of which is attached to its own motor, and a third wheel which passively rolls along preventing the robot from falling over. In this paper, we consider a stochastic version of a differential drive mobile robot, which captures actuator noise:

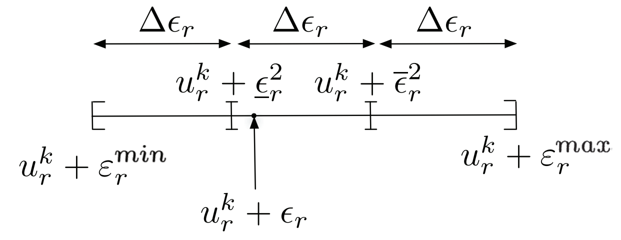
$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{r}{2}(u_r + \epsilon_r + u_l + \epsilon_l)\cos(\theta) \\ \frac{r}{2}(u_r + \epsilon_r + u_l + \epsilon_l)\sin(\theta) \\ \frac{r}{L}(u_r + \epsilon_r - u_l - \epsilon_l) \end{bmatrix}, \; u_r \in U_r, \; u_l \in U_l \tag{1}$$

where $(x,y) \in \mathbb{R}^2$ and $\theta \in [0, 2\pi)$ are the position and orientation of the vehicle in a world frame, u_r and u_l are the control inputs (angular velocities before being corrupted by noise), U_r and U_l are control constraint sets, and ϵ_r and ϵ_l are random variables modeling the actuator noise with continuous probability density functions supported on the bounded intervals $[\epsilon_r^{\min}, \epsilon_r^{\max}]$ and $[\epsilon_l^{\min}, \epsilon_l^{\max}]$, respectively. L is the distance between the two wheels and r is the wheel radius. We denote the state of the system by $q = [x, y, \theta]^{\mathrm{T}} \in SE(2)$.

Motivated by the fact that the time optimal trajectories for the bounded velocity differential drive robots are composed only of turns in place and straight lines (Balkcom and Mason, 2000), we assume U_r and U_l are finite, but we make no assumptions on optimality. We define

$$W_i = \{u + \epsilon | u \in U_i, \epsilon \in [\epsilon_i^{\min}, \epsilon_i^{\max}]\}, \; i \in \{r, l\}$$

as the sets of applied control inputs, i.e. the sets of angular wheel velocities that are applied to the system in the



$\Delta\epsilon_r$
$\Delta\epsilon_r$
$\Delta\epsilon_r$
$u_r^k + \underline{\epsilon}_r^2$
$u_r^k + \overline{\epsilon}_r^2$
$u_r^k + \varepsilon_r^{min}$
$u_r^k + \varepsilon_r^{max}$
$u_r^k + \epsilon_r$

Fig. 1. Let $n_r = 3$, i.e. $[\epsilon_r^{\min}, \epsilon_r^{\max}]$ is partitioned into three noise intervals of length $\Delta\epsilon_r$, $\mathcal{E}_r = \{[\underline{\epsilon}_r^1, \overline{\epsilon}_r^1], [\underline{\epsilon}_r^2, \overline{\epsilon}_r^2], [\underline{\epsilon}_r^3, \overline{\epsilon}_r^3]\}$. Assume the applied control input at stage k is $u_r^k + \epsilon_r$, such that $\epsilon_r \in [\underline{\epsilon}_r^2, \overline{\epsilon}_r^2]$. Then, the incremental encoder r, at stage k, will return measured interval $[\underline{w}_r^k, \overline{w}_r^k] = [u_r^k + \underline{\epsilon}_r^2, u_r^k + \overline{\epsilon}_r^2]$.

presence of noise. We assume that time is uniformly discretized (partitioned) into stages (intervals) of length Δt, where stage k is from $(k-1)\Delta t$ to $k\Delta t$. The duration of the entire motion plan is finite and it is denoted $K\Delta t$ (later in this section we explain how K is determined). We denote the control inputs and the applied control inputs at stage k as $u_i^k \in U_i, i \in \{r, l\}$, and $w_i^k \in W_i, i \in \{r, l\}$, respectively.

Sensing model: We assume that the vehicle is equipped with two incremental encoders, each measuring the applied control input (i.e. the angular velocity corrupted by noise) of one of the wheels. Motivated by the fact that the angular velocity is considered constant inside the given observation stage (Ekekwe et al., 2007; Petrella et al., 2007), the applied controls are considered piecewise constant, i.e. $w_i : [(k-1)\Delta t, k\Delta t] \rightarrow W_i, i \in \{r, l\}$, are constant over each stage.

We denote the measurement resolution of an incremental encoder (Petrella et al., 2007) as $\Delta\epsilon_i$, $i \in \{r, l\}$. Given $\Delta\epsilon_i$ and $[\epsilon_i^{\min}, \epsilon_i^{\max}]$, $i \in \{r, l\}$, then the following holds: $\exists n_i \in \mathbb{Z}^+$ s.t. $n_i \Delta\epsilon_i = |\epsilon_i^{\max} - \epsilon_i^{\min}|$, $i \in \{r, l\}$. For more details see Section 9 where we also explain how to obtain the measurement resolutions and the probability density functions. Then, $[\epsilon_i^{\min}, \epsilon_i^{\max}]$ can be partitioned[1] into n_i noise intervals of length $\Delta\epsilon_i$: $[\underline{\epsilon}_i^{j_i}, \overline{\epsilon}_i^{j_i}], j_i = 1, \ldots, n_i, i \in \{r, l\}$. We denote the set of all noise intervals by $\mathcal{E}_i = \{[\underline{\epsilon}_i^1, \overline{\epsilon}_i^1], \ldots, [\underline{\epsilon}_i^{n_i}, \overline{\epsilon}_i^{n_i}]\}$, $i \in \{r, l\}$. At stage k, if the applied control input is $u_i^k + \epsilon_i$, the incremental encoder i will return the measured interval

$$[\underline{w}_i^k, \overline{w}_i^k] = [u_i^k + \underline{\epsilon}_i, u_i^k + \overline{\epsilon}_i]$$

where $\epsilon_i \in [\underline{\epsilon}_i, \overline{\epsilon}_i] \in \mathcal{E}_i$, $i \in \{r, l\}$. In Figure 1 we give an example. The pair of measured intervals at stage k, $([\underline{w}_r^k, \overline{w}_r^k], [\underline{w}_l^k, \overline{w}_l^k])$, returned by the incremental encoders, is denoted by $\mathbb{W}^k$.

Remark 1 . *Even though we focus on a differential drive mobile robot, the approach presented in this paper can be applied to any system where the motion of the vehicle is given as a stochastic kinematic model equipped with a limited accuracy sensor as long as the following assumptions hold: (1) the actuator noise is modeled as a random variable with a continuous probability density function supported on a bounded interval, (2) the system is equipped with a limited accuracy sensor such that the actuator's noise interval can be partitioned into subintervals of length $\Delta\epsilon$, where $\Delta\epsilon$ is the measurement resolution of the sensor, and (3) given the actuator and sensor models, we can define an overapproximated uncertainty region (see Section 6.2). For example, the presented method can be extended to a stochastic version of a car-like robot equipped with two limited accuracy incremental encoders (Fraichard and Mermond, 1998) or to a stochastic version of a Dubins vehicle equipped with a limited accuracy gyroscope (Cizelj and Belta, 2012).*

Environment model and specification: The vehicle moves in a planar environment in which a set of non-overlapping regions of interest, denoted by R, is present. Let Π be the set of propositions satisfied at the regions in the environment. One of these propositions, denoted by $\pi_u \in \Pi$, signifies that the corresponding regions are `unsafe`. In this work, the motion specification is expressed as a BLTL formula ϕ over Π:

$$\phi = \neg\pi_u \mathbf{U}^{\leq T_1}(\varphi_1 \wedge \neg\pi_u \mathbf{U}^{\leq T_2}(\varphi_2 \wedge \cdots \wedge \neg\pi_u \mathbf{U}^{\leq T_f}\varphi_f)) \quad (2)$$

$f \in \mathbb{Z}^+$, and φ_j, $\forall j \in \{1, \ldots, f\}$, is of the following form:

$$\varphi_j = \mathbf{G}^{\leq \tau_j^1}(\bigvee_{\pi \in \Pi_j^1} \pi) \vee \cdots \vee \mathbf{G}^{\leq \tau_j^{n_j}}(\bigvee_{\pi \in \Pi_j^{n_j}} \pi)$$

where $n_j \in \mathbb{Z}^+$, $\forall_{n=1,\ldots,n_j} \Pi_j^n \subset \Pi \setminus \pi_u$, $\forall_{n=1,\ldots,n_j} \tau_j^n \in \mathbb{R}^{\geq 0}$ and $T_j \in \mathbb{R}^{\geq 0}$.

Example 3.1 . *Consider the environment shown in Figure 2. Let $\Pi = \{\pi_u, \pi_p, \pi_t, \pi_d\}$, where $\pi_u, \pi_p, \pi_t, \pi_d$ label the `unsafe`, `pick-up`, `test`, and the `drop-off` regions, respectively. Let the motion specification be as follows:*

Start from an initial state q_{init} and reach a `pick-up` region within T_1 time units to pick up a load. After entering the `pick-up` region reach a `test` region within T_2 time units and stay in it at least τ_2 time units. Finally, after entering the `test` region reach a `drop-off` region within T_3 time units to drop off the load. Always avoid the `unsafe` regions.

The specification translates to BLTL formula ϕ:

$$\phi = \neg\pi_u \mathbf{U}^{\leq T_1}(\pi_p \wedge \neg\pi_u \mathbf{U}^{\leq T_2}(\mathbf{G}^{\tau_2}\pi_t \wedge \neg\pi_u \mathbf{U}^{\leq T_3}\pi_d)) \; \blacksquare \quad (3)$$

Note that, in the formulas given by the proposed fragment of BLTL (equation (2)) the propositions are classified into two nonintersecting sets according to whether they represent regions that must be reached or avoided (the unsafe regions). By introducing an extended set of propositions (e.g., as presented in Fainekos et al. (2009)), this limiting assumption can be lifted, and the presented approach can be extended to BLTL formulas where a region needs to be both reached and avoided.

For simplicity, we assume that the vehicle can precisely determine its initial state $q_{\text{init}} = [x_{\text{init}}, y_{\text{init}}, \theta_{\text{init}}]$, in a known

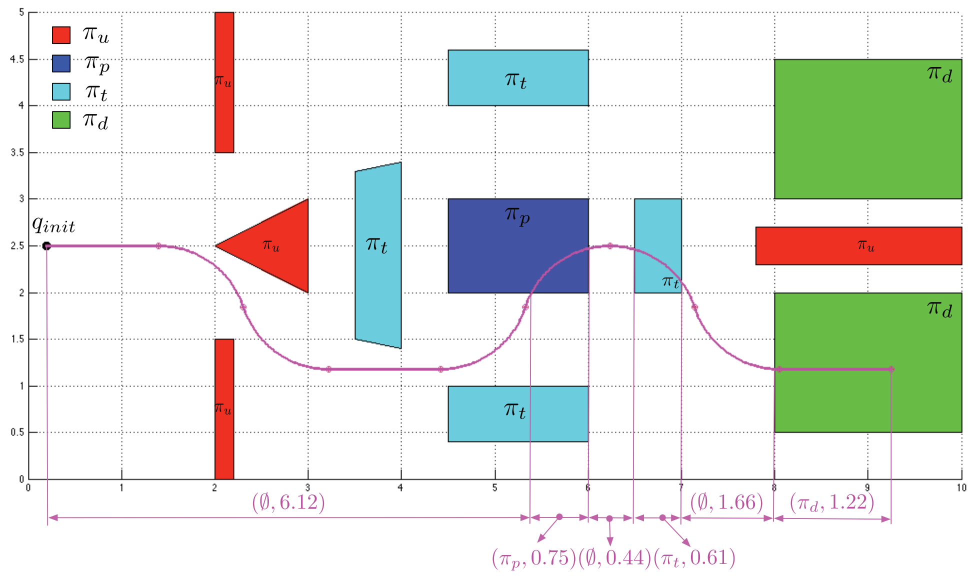


Fig. 2. An example environment with the regions of interest. The `unsafe`, `pick-up`, `test`, and the `drop-off` regions are shown in red, blue, cyan, and green, respectively. A sample state (position) trajectory of the system is shown in magenta.

map of the environment. However, our approach works as long as the initial state is such that the corresponding distance and orientation uncertainties are bounded (for more details see Section 6.2). For example, our approach can be applied directly when the initial state is modeled as a random variable with a finite support probability distribution (e.g. localization via a particle filter). While the vehicle moves, incremental encoder measurements $\mathbb{W}^k$ are available at each stage k. We define a *vehicle control strategy* as a map that takes as input a sequence of pairs of measured intervals $\mathbb{W}^1\mathbb{W}^2 \ldots \mathbb{W}^{k-1}$, and returns control inputs $u_r^k \in U_r$ and $u_l^k \in U_l$ at stage k. We are ready to formulate the main problem we consider in this paper:

Problem 1 . *Given a set of regions of interest R satisfying propositions from a set Π, a vehicle model described by equation (1) with initial state q_{init}, a motion specification expressed as a BLTL formula ϕ over Π (equation (2)), find a vehicle control strategy that maximizes the probability of satisfying the specification.*

To fully specify Problem 1, we need to define the satisfaction of a BLTL formula ϕ by a trajectory $q : [0, K\Delta t] \to SE(2)$ of the system from equation (1). A formal definition is included in Section 4. Informally, $q(t)$ produces a finite trace $\sigma = (o_1, t_1)(o_2, t_2)\ldots(o_l, t_l)$, $o_i \in \Pi \cup \emptyset$, $t_i \in \mathbb{R}^{\geq 0}$, $i \geq 1$, where o_i is the satisfied proposition[2] and t_i is the time spent satisfying o_i, as time evolves. A trajectory $q(t)$ satisfies BLTL formula ϕ if and only if the generated trace satisfies the formula. Given ϕ, for the duration of the entire motion plan we use the smallest $K \in \mathbb{Z}^+$ for which model checking a trace is well defined, i.e. the smallest K for which the maximum nested sum of time bounds (see Zuliani et al., 2010) is at most $K\Delta t$.

3.2. Approach

In this paper, we develop a suboptimal solution to Problem 1 consisting of three steps. First, we define a finite state MDP that captures every sequence realization of pairs of measurements returned by the incremental encoders. The states of the MDP correspond to the sequences of pairs of measured intervals and the actions correspond to the control inputs.

Second, we find a control policy for the MDP that maximizes the probability of satisfying BLTL formula ϕ. Because of the size of the MDP, finding the exact solution is computationally too expensive. We decided to trade-off correctness for scalability and we use a computationally efficient technique based on system sampling. We use recent results in SMC for MDPs (Henriques et al., 2012) to obtain an MDP control policy and a BIE algorithm (Zuliani et al., 2010) to estimate the probability of satisfying ϕ.

Finally, since each state of the MDP corresponds to a unique sequence of pairs of measured intervals, we translate the control policy to a vehicle control strategy. In addition, we show that the probability of satisfying ϕ, in the original environment, is bounded from below by the probability of satisfying the specification on the MDP under the obtained control policy.

Remark 2 . *The approach presented in this paper can only deal with finite time specifications. There are two main reasons for this. First, as it will be show in Section 6.2, the vehicle is subject to cumulative and unbounded position uncertainty. Second, in order to perform SMC, sampling and model checking of traces needs to be computationally feasible. Therefore, the reason why we decided to use BLTL was the fact that, unlike LTL or metric LTL (Koymans, 1990), the semantics of BLTL formulas is well defined* *on finite prefixes of traces with a duration that is bounded. Additionally, given a BLTL formula, we can determine the duration of the entire motion plan, $K\Delta t$, easily, as described in the previous subsection. Even though co-safe LTL (co-safe LTL formulas are LTL formulas such that any good trace satisfying the formula has a finite good prefix (see, for example, Bhatia et al., 2010)) can be used with our approach as well, it does not provide such a method for determining the duration of the entire motion plan. The reason for using the proposed fragment of BLTL is that it is the largest BLTL fragment for which we were able to prove the main result (Theorem 8.1), i.e. the fact that the probability that the vehicle satisfies the specification (given as the fragment of BLTL) in the environment is bounded from below by the probability of satisfying the specification on the corresponding MDP.*

4. Generating a trace

In this section, we formally define the satisfaction of a BLTL formula by a trajectory of (1). Let us denote $[\pi] = \{(x,y) \in \mathbb{R}^2 | (x,y) \in \cup_{r \in R_\pi} r\}$ as the set of positions that satisfy proposition π, where $R_\pi \subseteq R$ is the set of regions labeled with proposition π.

Definition 4.1 (Generating a trace). *The trace corresponding to a state trajectory $q(t) = [x(t), y(t), \theta(t)]^{\mathrm{T}}$ is a finite sequence $\sigma = (o_1, t_1)(o_2, t_2)\ldots(o_l, t_l)$, $o_i \in \Pi \cup \emptyset$, $t_i \in [0, K\Delta t]$, $i = 1, \ldots, l$, $l \geq 1$, where o_i is the satisfied proposition and t_i is the time spent satisfying o_i, generated according to the following rules, for all $t, t', \tau \in [0, K\Delta t]$:*

- $o_1 = \pi \in \Pi$ **iff** *(if and only if)* $(x(0), y(0)) \in [\pi]$ *and* $o_1 = \emptyset$ *otherwise.*
- *Let o_i be the satisfied proposition at some t. Then:*
  1. **If** $o_i = \emptyset$, **then** $o_{i+1} = \pi \in \Pi$, **iff** *(a)* $\exists t' > t$ *s.t.* $(x(t'), y(t')) \in [\pi]$, *and (b)* $\nexists \tau \in [t, t']$ *s.t.* $(x(\tau), y(\tau)) \in [\pi']$, $\forall \pi' \in \Pi$ **and** $t_i = \min_{t \in [\sum_{j=0}^{i-1} t_j, K\Delta t]} \{t | (x(t), y(t)) \in [\pi]\} - \sum_{j=0}^{i-1} t_j$, *with* $t_0 = 0$.
  2. **If** $o_i = \pi \in \Pi$, **then** $o_{i+1} = \emptyset$ **iff** $\exists t' > t$ *s.t.* $(x(t'), y(t')) \notin [\pi]$, **and** $t_i = \min_{t \in [\sum_{j=0}^{i-1} t_j, K\Delta t]} \{t | (x(t), y(t)) \notin [\pi]\} - \sum_{j=0}^{i-1} t_j$, *with* $t_0 = 0$.
- *Let for $K\Delta t$, o_l be the current satisfied propositions. Then, $t_l = K\Delta t - \sum_{j=1}^{l-1} t_j$.*

A trajectory $q(t)$ satisfies BLTL formula ϕ (equation (2)) if and only if the trace generated according to the rules stated above satisfies the formula. Note that, since the duration of the entire motion plan is finite, the generated trace is also finite. In Zuliani et al. (2010) the authors show that model checking a trace on a BLTL formula is well defined and can be based on only a finite prefix of the trace of bounded duration. The fact that the trace σ satisfies ϕ is denoted $\sigma \models \phi$. Given a trace σ, the ith state of σ, denoted σ_i, is (o_i, t_i), $i = 1, \ldots, l$. We denote $\sigma|_i$ as the finite subsequence of σ that starts in σ_i. Finally, given a formula ϕ in the form (2), we use ϕ_j to denote subformula $\neg\pi_u \mathbf{U}^{T_j} \varphi_j$, $j = 1, \ldots, f$. Using the BLTL semantics one can derive the following conditions to determine whether $\sigma \models \phi$:

Definition 4.2 (Satisfaction conditions). *Given a trace σ and a BLTL formula ϕ (equation (2)), let for $j \in \{1, \ldots, f\}$, $i_j, k_j \in \mathbb{N}$ be such that for some $n \in \{1, \ldots, n_j\}$ the following hold:*

1. $o_{i_j+k_j} \in \Pi_j^n$.
2. *For each* $i_j \leq i < i_j + k_j$, $o_i \neq \pi_u$.
3. $\sum_{i=i_j}^{i_j+k_j-1} t_i \leq T_j$.
4. $t_{i_j+k_j} \geq \tau_j^n$.

Then, $\sigma|_{i_j} \models \phi_j$. If $\forall j \in \{1, \ldots, f\}$, $\exists i_j, k_j \in \mathbb{N}$ s.t. $\sigma|_{i_j} \models \phi_j$ where $i_{j+1} = i_j + k_j$ with $i_1 = 1$, then $\sigma \models \phi$.

Example 4.3. *Consider the environment and the sample state (position) trajectory shown in Figure 2. Let ϕ be as in equation (3) with the following numerical values for the time bounds: $T_1 = 6.2$, $T_2 = 2.3$, $\tau_2 = 0.2$, and $T_3 = 2.3$. The trajectory generates trace $\sigma = (\emptyset, 6.12)(\pi_p, 0.75)(\emptyset, 0.44)(\pi_t, 0.61)(\emptyset, 1.66)(\pi_d, 1.22)$. The following holds: $\sigma|_1 \models \phi_1$ since for $i_1 = 1$ and $k_1 = 1$, $o_2 \in \{\pi_p\}$, $o_1 \neq \pi_u$, $t_1 \leq T_1$; $\sigma|_2 \models \phi_2$ since for $i_2 = 2$ and $k_2 = 2$, $o_4 \in \{\pi_t\}$, $o_2, o_3 \neq \pi_u$, $t_2 + t_3 \leq T_2$, and $t_4 \geq \tau_2$; and $\sigma|_4 \models \phi_3$ since for $i_3 = 4$ and $k_3 = 2$, $o_6 \in \{\pi_d\}$, $o_4, o_5 \neq \pi_u$, and $t_4 + t_5 \leq T_3$. Thus, $\sigma \models \phi$.* ■

5. Construction of an MDP model

Recall that ϵ_i is a random variable with a continuous probability density function supported on the bounded interval $[\epsilon_i^{\min}, \epsilon_i^{\max}]$, $i \in \{r, l\}$. The probability density functions are obtained through experimental trials (see Section 9) and they are defined as follows:

$$\Pr(\epsilon_i \in [\underline{\epsilon}_i^{j_i}, \overline{\epsilon}_i^{j_i}]) = p_i^{j_i} \tag{4}$$

$[\underline{\epsilon}_i^{j_i}, \overline{\epsilon}_i^{j_i}] \in \mathcal{E}_i, j_i = 1, \ldots, n_i$, s.t. $\sum_{j_i=1}^{n_i} p_i^{j_i} = 1, i \in \{r, l\}$.

An MDP M that captures every sequence realization of pairs of measurements returned by the incremental encoders is defined as a tuple $M = (S, s_0, Act, A, P)$, where:

- $S = \cup_{k=1,\ldots,K} \{([u_r + \underline{\epsilon}_r, u_r + \overline{\epsilon}_r], [u_l + \underline{\epsilon}_l, u_l + \overline{\epsilon}_l]) | u_r \in U_r, u_l \in U_l, [\underline{\epsilon}_r, \overline{\epsilon}_r] \in \mathcal{E}_r, [\underline{\epsilon}_l, \overline{\epsilon}_l] \in \mathcal{E}_l\}^k$. The meaning of the state is as follows: $(\mathbb{W}^1, \ldots, \mathbb{W}^k) \in S$, means that at stage i, $1 \leq i \leq K$, the pair of measured intervals is $\mathbb{W}^i$.
- $s_0 = \emptyset$ is the initial state.
- $\mathrm{Act} = U_r \times U_l \cup \xi$ is the set of actions, where ξ is a dummy action.
- $A : S \rightarrow 2^{\mathrm{Act}}$ gives the enabled actions at state s: if $|s| = K$, i.e. if the termination time is reached, $A(s) = \xi$, otherwise $A(s) = U_r \times U_l$.
- $P : S \times \mathrm{Act} \times S \rightarrow [0, 1]$ is a transition probability function constructed by the following rules:

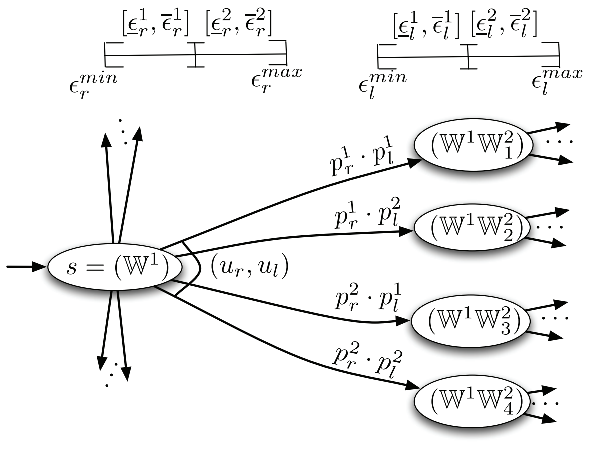

$[\underline{\epsilon}_r^1, \overline{\epsilon}_r^1]$ $[\underline{\epsilon}_r^2, \overline{\epsilon}_r^2]$
ϵ_r^{min}
ϵ_r^{max}
$[\underline{\epsilon}_l^1, \overline{\epsilon}_l^1]$ $[\underline{\epsilon}_l^2, \overline{\epsilon}_l^2]$
ϵ_l^{min}
ϵ_l^{max}
$s = (\mathbb{W}^1)$
(u_r, u_l)
$p_r^1 \cdot p_l^1$
$p_r^1 \cdot p_l^2$
$p_r^2 \cdot p_l^1$
$p_r^2 \cdot p_l^2$
$(\mathbb{W}^1\mathbb{W}_1^2)$
$(\mathbb{W}^1\mathbb{W}_2^2)$
$(\mathbb{W}^1\mathbb{W}_3^2)$
$(\mathbb{W}^1\mathbb{W}_4^2)$

Fig. 3. A fragment of the MDP M where $n_r = n_l = 2$. Thus, $p_r^m = \Pr(\epsilon_r \in [\underline{\epsilon}_r^m, \overline{\epsilon}_r^m])$, for $m = 1, 2$, and $p_l^n = \Pr(\epsilon_l \in [\underline{\epsilon}_l^n, \overline{\epsilon}_l^n])$, for $n = 1, 2$. Action $(u_r, u_l) \in A(s)$ enables four transitions. For example, given state $s = (\mathbb{W}^1)$, the new state is $(\mathbb{W}^1\mathbb{W}_2^2)$, where $\mathbb{W}_2^2 = ([u_r - \underline{\epsilon}_r^1, u_r + \overline{\epsilon}_r^1], [u_l - \underline{\epsilon}_l^2, u_l + \overline{\epsilon}_l^2])$, with probability $p_r^1 \cdot p_l^2$. This corresponds to applied control inputs being equal to $u_r + \epsilon_r$ and $u_l + \epsilon_l$ where $\epsilon_r \in [\underline{\epsilon}_r^1, \overline{\epsilon}_r^1]$ and $\epsilon_l \in [\underline{\epsilon}_l^2, \overline{\epsilon}_l^2]$.

1. If $s = (\mathbb{W}^1, \ldots, \mathbb{W}^k) \in S$ then $P(s, a, s') = p_r^m p_l^n$ iff $s' = (\mathbb{W}^1, \ldots, \mathbb{W}^k, ([u_r + \underline{\epsilon}_r^m, u_r + \overline{\epsilon}_r^m], [u_l + \underline{\epsilon}_l^n, u_l + \overline{\epsilon}_l^n])) \in S$ and $a = (u_r, u_l) \in \{U_r \times U_l\}$ where $m = 1, \ldots, n_r$, $n = 1, \ldots, n_l$ and $k = 1, \ldots, K$.
2. If $|s| = K$ then $P(s, a, s') = 1$ iff $a = \xi$ and $s' = s$.
3. $P(s, a, s') = 0$ otherwise.

Rule (1) defined above follows from the fact that given u_r^k and u_l^k as the control inputs at stage k, the pair of measured intervals at stage $k+1$ is $([u_r^k + \underline{\epsilon}_r^m, u_r^k + \overline{\epsilon}_r^m], [u_l^k + \underline{\epsilon}_l^n, u_l^k + \overline{\epsilon}_l^n])$ with probability $p_r^m p_l^n$, since $\Pr(\epsilon_r \in [\underline{\epsilon}_r^m, \overline{\epsilon}_r^m]) = p_r^m$ and $\Pr(\epsilon_l \in [\underline{\epsilon}_l^n, \overline{\epsilon}_l^n]) = p_r^n$, which follows from equation (4) (see the MDP fragment in Figure 3). Rule (2) states that if the length of s is equal to K, i.e. if the termination time is reached, then $A(s) = \xi$ with $P(s, \xi, s) = 1$.

Proposition 5.1. *The model M defined above is a valid MDP, i.e. it satisfies the Markov property and P is a transition probability function.*

Proof. The proof follows from construction of P. Given current state $s \in S$ and an action $a \in A(s)$, the conditional probability distribution of future states depends only on the current state s, not on the sequences of events that preceded it (see rule (1) above). Thus, the Markov property holds. In addition, since for every s and $a \in A(s)$: $\sum_{s' \in S} P(s, a, s') = \sum_{m=1}^{n_r} \sum_{n=1}^{n_l} p_r^m p_l^n = \sum_{m=1}^{n_r} p_r^m \sum_{n=1}^{n_l} p_l^n = 1$, it follows that P is a valid transition probability function. ■

6. Position uncertainty

6.1. Nominal state trajectory

For each interval belonging to the set of noise intervals $\mathcal{E}_i$, we define a representative value $\epsilon_i^{j_i} = (\underline{\epsilon}_i^{j_i} + \overline{\epsilon}_i^{j_i})/2$, $j_i = 1, \ldots, n_i$, $i \in \{r, l\}$, i.e. $\epsilon_i^{j_i}$ is the midpoint of interval $[\underline{\epsilon}_i^{j_i}, \overline{\epsilon}_i^{j_i}] \in \mathcal{E}_i$, $i \in \{r, l\}$. We denote the set of representative values as $E_i = \{\epsilon_i^1, \ldots, \epsilon_i^{n_i}\}$, $i \in \{r, l\}$.

We use $q^k(t)$, w_r^k, and w_l^k, $t \in [(k-1)\Delta t, k\Delta t]$, $k = 1, \ldots, K$, to denote the state trajectory and the constant applied controls at stage k, respectively. With a slight abuse of notation, we use q^k to denote the end of state trajectory $q^k(t)$, i.e. $q^k = q^k(k\Delta t)$. Given state q^{k-1}, the state trajectory $q^k(t)$ can be derived by integrating the system given by equation (1) from the initial state q^{k-1}, and taking into account that the applied controls are constant and equal to w_r^k and w_l^k. Throughout the paper, we will also denote this trajectory by $q^k(q^{k-1}, w_r^k, w_l^k, t)$, when we want to explicitly capture the initial state q^{k-1} and the constant applied controls w_r^k and w_l^k.

Given a path through the MDP:

$$s_0 \xrightarrow{(u_r^1, u_l^1)} s_1 \xrightarrow{(u_r^2, u_l^2)} s_2 \cdots s_{K-1} \xrightarrow{(u_r^K, u_l^K)} s_K \qquad (5)$$

where $s_k = (\mathbb{W}^1, \ldots, \mathbb{W}^k)$, with $\mathbb{W}^k = ([u_r^k + \underline{\epsilon}_r^k, u_r^k + \overline{\epsilon}_r^k], [u_l^k + \underline{\epsilon}_l^k, u_l^k + \overline{\epsilon}_l^k])$, $k = 1, \ldots, K$, we define the *nominal state trajectory* $q(t)$, $t \in [0, K\Delta t]$, as follows:

$$q(t) = q^k(q^{k-1}, u_r^k + \epsilon_r^k, u_l^k + \epsilon_l^k, t), \ t \in [(k-1)\Delta t, \Delta t]$$

$k = 1, \ldots, K$, where $\epsilon_i^k \in E_i$ is such that $\epsilon_r^k \in [\underline{\epsilon}_i^k, \overline{\epsilon}_i^k]$, $i \in \{r, l\}$ and $q^0 = q_{\text{init}}$. For every path through the MDP, its nominal state trajectory is well defined. The next step is to define the uncertainty evolution, along the nominal state trajectory, since the applied controls can take any value within the measured intervals.

6.2. Position uncertainty evolution

Since a motion specification is a statement about the propositions satisfied by the regions of interest in the environment, in order to answer whether some state trajectory satisfies BLTL formula ϕ it is sufficient to know its projection in $\mathbb{R}^2$. Therefore, we focus only on the position uncertainty.

The position uncertainty of the vehicle when its nominal position is $(x, y) \in \mathbb{R}^2$ is modeled as a disc centered at (x, y) with radius $d \in \mathbb{R}$, where d denotes the distance uncertainty:

$$D((x, y), d) = \{(x'y') \in \mathbb{R}^2 | ||(x, y), (x', y')|| \leq d\} \qquad (6)$$

where $|| \cdot ||$ denotes the Euclidian distance. Next, we explain how to obtain d.

First, let $\Delta\theta \in S^1$ denote the orientation uncertainty. Let $q(t)$, $t \in [0, K\Delta t]$, be the nominal state trajectory corresponding to a path through the MDP (equation (5)). Then, $q(t)$ can be partitioned into K state trajectories: $q^k(t) = q^k(q^{k-1}, u_r^k + \epsilon_r^k, u_l^k + \epsilon_l^k, t)$, $t \in [(k-1)\Delta t, \Delta t]$, $k = 1, \ldots, K$, where $\epsilon_i^k \in E_i$ is such that $\epsilon_i^k \in [\underline{\epsilon}_i^k, \overline{\epsilon}_i^k] \in \mathcal{E}_i$, $i \in \{r, l\}$ and $q^0 = q_{\text{init}}$ (see Figure 4). The distance and orientation uncertainty at state q^k are denoted as d^k and $\Delta\theta^k$, respectively. We set d^k and $\Delta\theta^k$ at state $q^k = [x^k, y^k, \theta^k]^T$ equal to:

$$d^k = \max_{[x',y',\theta']^T \in \mathcal{R}^k}(\,||(x^k, y^k), (x', y')||) + d^{k-1} \text{ and}$$
$$\Delta\theta^k = \max_{[x',y',\theta']^T \in \mathcal{R}^k}(\,|\theta^k - \theta'|) \tag{7}$$

where

$$\mathcal{R}^k = \{q^k([x^{k-1}, y^{k-1}, \theta^{k-1} + \alpha]^T, u_r^k + \epsilon_r', u_l^k + \epsilon_l', k\Delta t) \mid \alpha \in \{\Delta\theta^{k-1}, -\Delta\theta^{k-1}\}, \epsilon_r' \in \{\underline{\epsilon}_r^k, \overline{\epsilon}_r^k\}, \epsilon_l' \in \{\underline{\epsilon}_l^k, \overline{\epsilon}_l^k\}\} \tag{8}$$

for $k = 1, \ldots, K$, where $d^0 = 0$ and $\Delta\theta^0 = 0$.

Equations (7) and (8) are obtained using a worst-case scenario assumption. At stage k, the pair of measured intervals is $\mathbb{W}^k = ([u_r^k + \underline{\epsilon}_r^k, u_r^k + \overline{\epsilon}_r^k], [u_l^k + \underline{\epsilon}_l^k, u_l^k + \overline{\epsilon}_l^k])$ and we use the endpoints of the measured intervals to define set $\mathcal{R}^k$. $\mathcal{R}^k$ is the smallest set of points in $SE(2)$, at the end of stage k, guaranteed to contain (a) the state with the maximum distance (in Euclidian sense) from q^k given that the applied controls at stage i are within the measured intervals at stage i, and (b) the state with the maximum orientation difference compared to q^k given that the applied controls at stage i are within the measured intervals at stage i, $i = 1, \ldots, k$. (For more details about $\mathcal{R}^k$ see Fraichard and Mermond (1998).) An example is given in Figure 4.

From equations (7) and (8) it follows that, given a nominal state trajectory $q(t)$, $t \in [0, K\Delta t]$, the distance uncertainty increases as a function of time. The way it changes along $q(t)$ makes it difficult to characterize the exact shape of the position uncertainty region. Instead, we use a conservative approximation of the region. We define $d : [0, K\Delta t] \to \mathbb{R}$ as an *approximate distance uncertainty trajectory* and we set $d(t) = d^k$, $t \in [(k-1)\Delta t, k\Delta t]$, $k = 1, \ldots, K$, i.e. we set the distance uncertainty along the state trajectory $q^k(t)$ equal to the maximum value of the distance uncertainty along $q^k(t)$, which is at state q^k. An example illustrating this idea is given in Figure 4.

Proposition 6.1 . *Given a path through the MDP M (equation (5)), and the corresponding $q(t)$ and $d(t)$, $t \in [0, K\Delta t]$, as defined above, then any state trajectory $q'(t) = q^k(q^{k-1}, u_r^k + \epsilon_r^{k'}, u_l^k + \epsilon_l^{k'}, t)$, $t \in [(k-1)\Delta t, k\Delta t]$, $k = 1, \ldots, K$, where $q^0 = q_{\text{init}}$, $\epsilon_r^{k'} \in [\underline{\epsilon}_r^k, \overline{\epsilon}_r^k]$ and $\epsilon_l^{k'} \in [\underline{\epsilon}_l^k, \overline{\epsilon}_l^k]$, is within the uncertainty region, i.e. $(x'(t), y'(t)) \in D((x(t), y(t)), d(t))$, $\forall t \in [0, K\Delta t]$.*

Proof: The proof follows from the definition of the approximate distance uncertainty trajectory and equations (6)–(8). ■

7. Generating a trace under the position uncertainty

Let $q(t)$ be a nominal state trajectory with the distance uncertainty trajectory $d(t)$, $t \in [0, K\Delta t]$. In this subsection we introduce a set of conservative rules according to which the trace corresponding to the uncertainty region $D((x(t), y(t)), d(t))$ is generated. These rules guarantee that if the generated trace satisfies ϕ (equation (2)) then any state (position) trajectory inside $D((x(t), y(t)), d(t))$ will satisfy ϕ.

Definition 7.1 (Generating a trace under uncertainty). *The trace corresponding to an uncertainty region $D((x(t), y(t)), d(t))$ is a finite sequence $\sigma = (o_1, t_1)(o_2, t_2), \ldots, (o_l, t_l)$, $o_i \in \Pi \cup \emptyset$, $t_i \in [0, K\Delta t]$, $i = 1, \ldots, l$, $l \geq 1$, where o_i is the satisfied proposition and t_i is the time spent satisfying o_i, generated according to the following rules, for all $t, t', \tau \in [0, K\Delta t]$:*

- *$o_1 = \pi \in \Pi \setminus \pi_u$* **iff** *$D((x(0), y(0), d(0)) \subseteq [\pi]$, $o_1 = \pi_u$* **iff** *$D((x(0), y(0), d(0)) \cap [\pi_u] \neq \emptyset$ and $o_1 = \emptyset$ otherwise.*
- *Let o_i be the satisfied proposition at some t. Then:*
  1. **If** *$o_i = \pi \in \Pi \setminus \pi_u$,* **then** *$o_{i+1} = \emptyset$* **iff** *$\exists t' > t$ s.t. $D((x(t'), y(t')), d(t')) \nsubseteq [\pi]$* **and** *$t_i = \min_{t \in [\sum_{j=0}^{i-1} t_j, K\Delta t]}\{t | D((x(t), y(t)), d(t)) \nsubseteq [\pi]\} - \sum_{j=0}^{i-1} t_j$, with $t_0 = 0$.*
  2. **If** *$o_i = \pi_u$,* **then** *$o_{i+1} = \emptyset$* **iff** *$\exists t' > t$ s.t. $D((x(t'), y(t')), d(t')) \cap [\pi_u] = \emptyset$* **and** *$t_i = \min_{t \in [\sum_{j=0}^{i-1} t_j, K\Delta t]}\{t | D((x(t), y(t)), d(t)) \cap [\pi_u] = \emptyset\} - \sum_{j=0}^{i-1} t_j$, with $t_0 = 0$.*
  3. **If** *$o_i = \emptyset$,* **then** *$o_{i+1} = \pi \in \Pi \setminus \pi_u$,* **iff**
     (a) $\exists t' > t$ s.t. $D((x(t'), y(t')), d(t')) \subseteq [\pi]$
     (b) $\nexists \tau \in [t, t']$ s.t. $D((x(\tau), y(\tau)), d(\tau)) \subseteq [\pi']$, $\forall \pi' \in \Pi \setminus \pi_u$
     (c) $\nexists \tau \in [t, t']$ s.t. $D((x(\tau), y(\tau)), d(\tau)) \cap [\pi_u] \neq \emptyset$

     and *$t_i = \min_{t \in [\sum_{j=0}^{i-1} t_j, K\Delta t]}\{t | D((x(t), y(t)), d(t)) \subseteq [\pi]\} - \sum_{j=0}^{i-1} t_j$, with $t_0 = 0$.*
  4. **If** *$o_i = \emptyset$,* **then** *$o_{i+1} = \pi_u$,* **iff**
     (a) $\exists t' > t$ s.t. $D((x(t'), y(t')), d(t')) \cap [\pi_u] \neq \emptyset$
     (b) $\nexists \tau \in [t, t']$ s.t. $D((x(\tau), y(\tau)), d(\tau)) \subseteq [\pi']$, $\forall \pi' \in \Pi \setminus \pi_u$

     and *$t_i = \min_{t \in [\sum_{j=0}^{i-1} t_j, K\Delta t]}\{t | D((x(t), y(t)), d(t)) \cap [\pi_u] \neq \emptyset\} - \sum_{j=0}^{i-1} t_j$, with $t_0 = 0$.*
- *For $K\Delta t$, let o_l be the current satisfied proposition. Then $t_l = K\Delta t - \sum_{j=1}^{l-1} t_j$.*

In Figure 5 we show an uncertainty region and the corresponding trace generated according to the rules stated above. Next, we show that if the trace corresponding to an uncertainty region satisfies ϕ, then any state (position) trajectory inside the uncertainty region also satisfies ϕ.

Proposition 7.2 . *Let $D((x(t), y(t)), d(t))$ be the uncertainty region corresponding to a path through the MDP M (equation (5)) and let $q'(t)$ be any state trajectory as defined in Proposition 6.1. Let $\sigma^D = (o_1^D, t_1^D) \ldots (o_k^D, t_k^D)$*

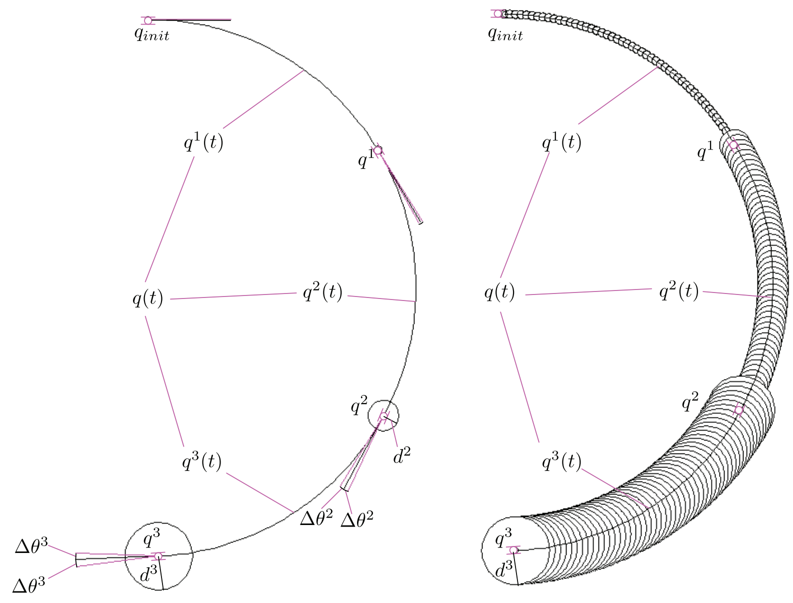


Fig. 4. *Left*: Evolution of the position uncertainty along the nominal state trajectory $q(t) = [x(t), y(t), \theta(t)]$, where $q(t)$ is partitioned into three state trajectories, $q^k(t)$, $k = 1, 2, 3$. *Right*: The conservative approximation of region $D((x(t), y(t)), d(t))$ along $q(t)$, where the distance uncertainty trajectory is $d(t') = d^k(t)$, $t' \in [(k-1)\Delta t, k\Delta t]$, where $d^k(t) = d^k$, $k = 1, 2, 3$.

and $\sigma^{q'} = (o_1^{q'}, t_1^{q'}) \dots (o_l^{q'}, t_l^{q'})$ *be the corresponding traces. Given BLTL formula* ϕ *(equation (2)), if* $\sigma^D \vDash \phi$, *then* $\sigma^{q'} \vDash \phi$.

Proof. First, we state two relations between the given traces:

1. Let $o_i^D = \pi \in \Pi \setminus \pi_u$ for some $i \in \{1, \dots, k\}$.
   Then, the following holds: $\exists j \in \{1, \dots, l\}$ such that $o_j^{q'} = \pi$ and $t_i^D \leq t_j^{q'}$.
   Informally, if t_i^D is the time $D((x(t), y(t)), d(t))$ spent inside the region satisfying proposition π, then $q'(t)$ will spend at least t_i^D time units inside that region.
2. Let $o_i^D = \pi \in \Pi \setminus \pi_u$ and $o_{i'}^D = \pi' \in \Pi \setminus \pi_u$ for some $i, i' \in \{1, \dots, k\}$, $i' > i$. Then, the following holds: $\exists j, j' \in \{1, \dots, l\}, j' > j$ such that $o_j^{q'} = \pi$ and $o_{j'}^{q'} = \pi'$. In addition, $\sum_{h=j}^{j'-1} t_h^{q'} \leq \sum_{h=i}^{i'-1} t_h^D$.

   Informally, if the time between $D((x(t), y(t)), d(t))$ entering a region satisfying π and then entering a region satisfying π' is $\sum_{h=i}^{i'-1} t_h^D$ time units, then the time between $q'(t)$ entering the region satisfying π and then entering the region satisfying π' is bounded from above by $\sum_{h=i}^{i'-1} t_h^D$. For more intuition about these relations see Figure 5.

Assuming $\sigma^D \vDash \phi$, then $\forall j \in \{1, \dots, f\}$, $\exists i_j, k_j \in \mathbb{N}$ and some $n \in \{1, \dots, n_j\}$ such that $\sigma_{i_j}^D \vDash \phi_j$ (see Definition 4.2). Then, from Proposition 6.1 and Definition 4.1 and 7.1, it follows that $\forall j \in \{1, \dots, f\}$, $\exists s_j, z_j \in \mathbb{N}$ such that:

1. $o_{s_j+z_j}^{q'} \in \Pi_j^n$.
2. For each $s_j \leq i < s_j + z_j$, $o_i^{q'} \neq \pi_u$.
3. $\sum_{i=s_j}^{s_j+z_j-1} t_i^{q'} \leq \sum_{i=i_j}^{i_j+k_j-1} t_i^D \leq T_j$ (2nd relation above).
4. $t_{s_j+z_j}^{q'} \geq t_{i_j+k_j}^D \geq \tau_j^n$ (1st relation above).

where $s_{j+1} = s_j + z_j$ with $s_1 = 1$.

Thus, $\forall j \in \{1, \dots, f\}$, $\sigma_{s_j}^{q'} \vDash \phi_j$, and according to Definition 4.2, it follows that $\sigma^{q'} \vDash \phi$. In Figure 5 we give an example. ■

8. Vehicle control strategy

Given the MDP M, the next step is to obtain a control policy that maximizes the probability of generating a path through M such that the corresponding trace (as defined in Sections 6 and 7) is satisfying. There exist approaches that, given an MDP and a temporal logic formula, generate an exact control policy that maximizes the probability of satisfying the specification. In general, exact techniques rely on reasoning about the entire state space, which is a limiting factor in their applicability to large problems.

Given U_r, U_l, n_r, n_l, and K, the size of the MDP M is bounded above by $(|U_r| \times |U_l| \times n_r \times n_l)^K$. Even for a simple case study, due to the size of M, using the exact methods to obtain a control policy is computationally too expensive. Therefore, we decide to trade-off correctness for scalability and use computationally efficient techniques based on system sampling. In particular, we modify SMC for MDPs (Henriques et al., 2012) that selectively samples traces of an MDP until enough statistical evidence has been found to support the claim that some property holds in the MDP with some probability. The problem is reduced to finding the probability under an optimal control policy: one that maximizes the probability of satisfying the property. The

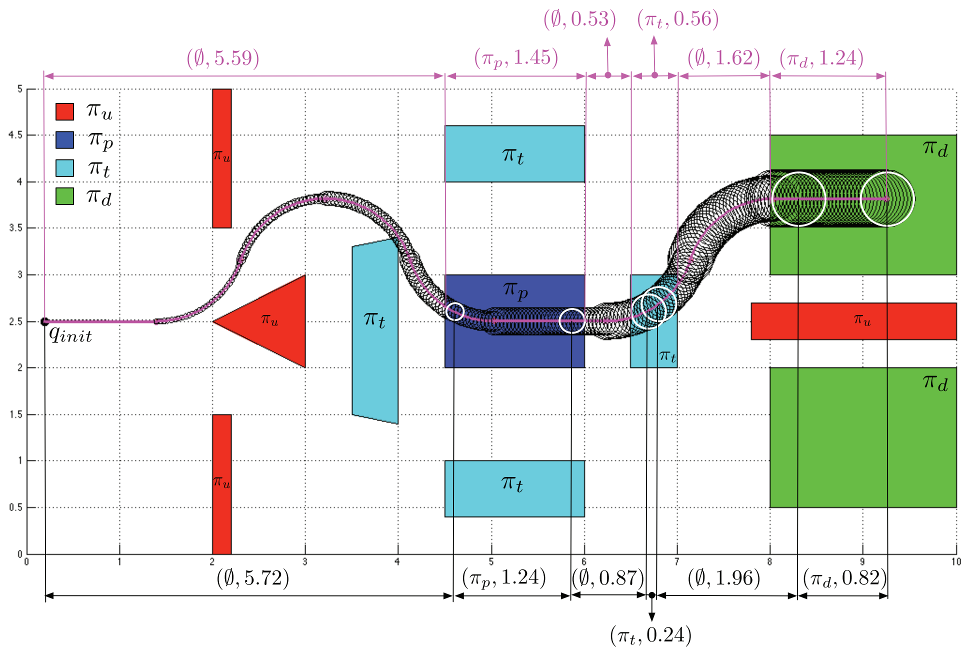


Fig. 5. An uncertainty region and a sample state (position) trajectory, inside the uncertainty region, are shown in black and magenta, respectively. The corresponding generated traces are σ^D=$(\emptyset, 5.72)(\pi_p, 1.24)(\emptyset, 0.87)(\pi_t, 0.24)(\emptyset, 1.96)(\pi_d, 0.82)$ and $\sigma^{q'} = (\emptyset, 5.59)(\pi_p, 1.45)(\emptyset, 0.53)(\pi_t, 0.56)(\emptyset, 1.62)(\pi_d, 1.24)$. Let ϕ be as given in Example 4.3. Then, it follows that $\sigma^D \models \phi$ and $\sigma^{q'} \models \phi$. Note that for $\sigma_2^D = \sigma_2^{q'} = \pi_p$, $t_2^D < t_2^{q'}$ (1st relation above). Also, for $\sigma_2^D = \sigma_2^{q'} = \pi_p$ and $\sigma_4^D = \sigma_4^{q'} = \pi_t$, $\sum_{i=2}^{3} t_i^{q'} < \sum_{i=2}^{3} t_i^D$ (2nd relation above).

optimal control policy returned by the approach is the solution to our original problem, and since the approach is sample based it considers only a very small fraction of potential control policies.

8.1. Overview

We obtain a suboptimal control policy by iterating over the *control synthesis* and the *probability estimation* procedure until the stopping criterion is met (see Section 8.3). In the control synthesis procedure we use the control synthesis approach from Henriques et al. (2012) to generate a control policy for the MDP M. In particular we use a *control policy optimization* part of the algorithm which consists of the *control policy evaluation* and the *control policy improvement* procedure to incrementally improve a candidate control policy (control policy is initialized with a uniform distribution at each state). Next, in the probability estimation procedure we use SMC by BIE, as presented in Zuliani et al. (2010). We estimate the probability that the MDP M, under the candidate control policy, generates a path such that the corresponding trace satisfies BLTL formula ϕ. Finally, if the estimated probability converges, i.e. if the stopping criterion is met, we map the control policy to a vehicle control strategy. Otherwise, the control synthesis procedure is restarted using the latest update of the control policy. The flow of this approach is depicted in Figure 6.

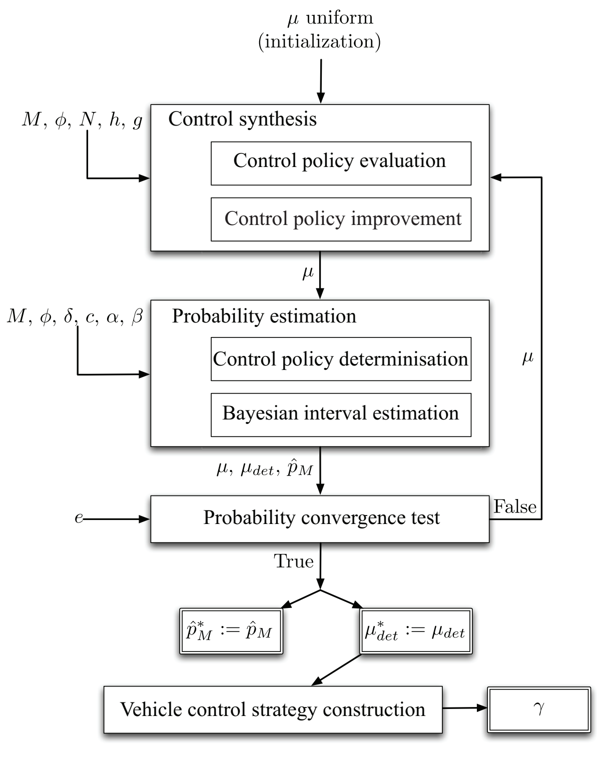

μ uniform (initialization)
M, ϕ, N, h, g
Control synthesis
Control policy evaluation
Control policy improvement
μ
M, ϕ, δ, c, α, β
Probability estimation
Control policy determinisation
Bayesian interval estimation
μ
μ, μ_{det}, $\hat{p}_M$
e
Probability convergence test
False
True
$\hat{p}^*_M := \hat{p}_M$
$\mu^*_{det} := \mu_{det}$
Vehicle control strategy construction
γ

Fig. 6. Flow chart of the approach used to obtain the vehicle control strategy.

Remark 3 . *In general, in order to use model checking tools for MDP control synthesis from temporal logic motion specifications, the MDP needs to contain the information about the motion of the vehicle in the environment. Note that, the MDP M defined in Section 5 only captures the sequences of measurements returned by the incremental encoders, and it does not explicitly capture the motion of the vehicle in the environment. However, given M and a motion specification expressed as a BLTL formula, the proposed control synthesis approach returns a control policy for M. This is due to the fact that SMC for MDPs allows for, first, sampling a path through M, and only then generating the corresponding trace (i.e. relating the path to the motion of the vehicle in the environment) and model checking it.*

8.2. Control synthesis

The details of the control policy optimization algorithm can be found in Henriques et al. (2012), and here we only give an informal overview of the approach. In the control policy evaluation procedure we sample paths of the MDP M under the current control policy μ. Given a path $\omega = s_0 \xrightarrow{a^1} s_1 \xrightarrow{a^2} s_2 \cdots s_{K-1} \xrightarrow{a^K} s_K$, where $a^k = (u_r^k, u_l^k)$, the corresponding trace σ is generated as described in Sections 6 and 7. Next, we check formula ϕ on each σ and estimate how likely it is for each action to lead to the satisfaction of BLTL formula ϕ, i.e. we obtain the estimate of the probability that a path crossing a state–action pair, (s^k, a^{k+1}), $k = 0, \ldots, K-1$, in ω will generate a trace that satisfies ϕ. These estimates are then used in the control policy improvement procedure, in which we update the control policy μ by reinforcing the actions that led to the satisfaction of ϕ most often. In Henriques et al. (2012), the authors show that the updated control policy is provably better than the previous one by focusing on the more promising regions of the state space.

The algorithm takes as input MDP M, BLTL formula ϕ and the current control policy μ, together with the parameters of the algorithm (a greediness parameter $0 < g < 1$, a history parameter $0 < h < 1$, and the number of sample paths in control policy evaluation procedure, denoted by N), and returns the updated probabilistic control policy μ.

Despite being sufficient to achieve maximum probabilities, deterministic control policies are a poor choice for exploring the state space through simulation. Therefore, in the control synthesis procedure we always use probabilistic control policies since they are more flexible and enable reinforcement of different actions. However, in the probability estimation procedure, we use deterministic control policies in order to redirect the residual probabilities of choosing bad actions to the promising regions of the state space. Thus, in the next stage we use the deterministic version of μ, denoted μ_{det}, where for all $s \in S$ and $a \in A$,

$$\mu_{\text{det}}(s,a) = \begin{cases} 1 & \text{if } a = \text{Rand}(\arg\max_{a \in Act(s)} \mu(s,a)) \\ 0 & \text{Otherwise} \end{cases}$$

where $\text{Rand}(X) \rightarrow x \in X$ is a function that given a nonempty set X returns an element $x \in X$ with probability $\frac{1}{|X|}$. In words, we compute a control policy that always picks the best estimated action at each state. If at a state there are multiple actions that achieve the maximum, function Rand returns, with equal probability, one of the maximizing actions.

8.3. Probability estimation

Next, we determine the estimate of the probability that the MDP M, under the deterministic control policy μ_{det}, generates a path such that the corresponding trace satisfies BLTL formula ϕ. To do so we use the BIE algorithm as presented in Zuliani et al. (2010). We denote the exact probability as p_M and the estimate as $\hat{p}_M$.

The inputs of the algorithm are MDP M, control policy μ_{det}, BLTL formula ϕ, half interval size $\delta \in (0, \frac{1}{2})$, interval coefficient $c \in (\frac{1}{2}, 1)$, and the coefficients α, β of the Beta prior. The algorithm returns $\hat{p}_M$. The algorithm generates traces by sampling paths through M under μ_{det} (as described in Sections 6 and 7) and checks whether the corresponding traces satisfy ϕ, until enough statistical evidence has been found to support the claim that p_M is inside the interval $[\hat{p}_M - \delta, \hat{p}_M + \delta]$ with arbitrarily high probability, i.e. $\Pr(p_M \in [\hat{p}_M - \delta, \hat{p}_M + \delta]) \geq c$.

We stop iterating over the control synthesis and the probability estimation procedure when the difference between the two consecutive probability estimates converges to a neighborhood of radius $e \in (0, 1)$, i.e. when the difference is smaller or equal to e. Let μ^*_{det} and $\hat{p}^*_M$ be the current control policy and the corresponding probability estimate, respectively, when the stopping criterion is met.

8.4. Control strategy

The vehicle control strategy is a function $\gamma : S \rightarrow U_r \times U_l$ that maps a sequence of pairs of measured intervals, i.e. a state of the MDP, to the control inputs:

$$\gamma((\mathbb{W}^1, \ldots, \mathbb{W}^k)) = \gamma(s_k) = \arg\max_{a \in Act(s_k)} \mu^*_{\text{det}}(s_k, a) \tag{9}$$

$k = 1, \ldots, K-1$ with $\gamma(s_0) = \arg\max_{a \in Act(s_0)} \mu^*_{\text{det}}(s_0, a)$

At stage k, the control inputs are

$$(u_r^k, u_l^k) = \gamma((\mathbb{W}^1, \ldots, \mathbb{W}^{k-1})) \in \{U_r \times U_l\}.$$

Thus, given a sequence of pairs of measured intervals, γ returns the control inputs for the next stage; the control inputs are equal to the action returned by μ^*_{det} at the state of the MDP corresponding to that sequence.

Theorem 8.1. *The probability that the system given by equation (1), under the vehicle control strategy γ, generates a state trajectory that satisfies BLTL formula ϕ (equation (2)) is bounded from below by p^*_M, where $Pr(p^*_M \in [\hat{p}^*_M - \delta, \hat{p}^*_M + \delta]) \geq c$.*

Proof: Let ω be a path through the MDP M and $D((x(t), y(t)), d(t))$ be the corresponding uncertainty region as defined in Section 6. The probability that the system given by equation (1), under γ, generates a state trajectory $q'(t)$ as defined in Proposition 6.1 is equal to the probability of generating path ω under μ^*_{det}. Let σ^D and $\sigma^{q'}$ be the corresponding traces. We consider two cases: (a) trace σ^D satisfies ϕ and (b) trace σ^D does not satisfy ϕ.

Let us first consider the former. If $\sigma^D \models \phi$ from Proposition 7.2 it follows that $\sigma^{q'} \models \phi$. Since under μ^*_{det} the probability that a path through the MDP M generates a satisfying trace is p^*_M, it follows that the probability that the system given by equation (1), under γ, will generate a satisfying state trajectory is also p^*_M. To show that p^*_M is the lower bound we need to consider the latter case. It is sufficient to observe that because of the conservative approximation of $D((x(t), y(t)), d(t))$ it is possible that $\sigma^{q'}$ satisfies ϕ, even though σ^D does not satisfy it. Therefore, it follows that the probability that the system given by equation (1), under the vehicle control strategy γ, generates a state trajectory that satisfies BLTL formula ϕ, is bounded from below by p^*_M. The rest of the proof, i.e. $\Pr(p^*_M \in [\hat{p}^*_M - \delta, \hat{p}^*_M + \delta]) \geq c$, is given in Zuliani et al. (2010). ■

8.5. *Complexity*

As stated above, the size of the MDP M is bounded above by $(|U_r| \times |U_l| \times n_r \times n_l)^K$. Obviously, it can be expensive (in sense of memory usage) to store the whole MDP. Since our approach is sample-based, it is not necessary for the MDP to be constructed explicitly. Instead, a state of the MDP is stored only if it is sampled during the control synthesis procedure. As a result, during the execution, the number of states stored in the memory is bounded above by $N \times K \times n$, where n is the number of iterations between the control synthesis and the probability estimation procedures. In the next section, through experiments and simulations, we provide further insight into the complexity of our method. The complexity analysis of the control synthesis part can be found in Henriques et al. (2012) and the complexity analysis of BIE algorithm can be found in Zuliani et al. (2010).

9. Case studies

We considered the system given by equation (1) and we used the numerical values corresponding to Dr Robot's x80Pro mobile robot (http://www.drrobot.com/products.asp) equipped with two incremental encoders. The parameters were $r = 0.085$ m and $L = 0.295$ m. To reduce the complexity, $U_r \times U_l$ was limited to $\{(\frac{1+L}{4r}, \frac{1-L}{4r}), (\frac{1}{4r}, \frac{1}{4r}), (\frac{1-L}{4r}, \frac{1+L}{4r})\}$, where the pairs of control inputs corresponded to a vehicle turning left at $\frac{1}{2}\frac{\text{rad}}{\text{s}}$, going straight, and turning right at $\frac{1}{2}\frac{\text{rad}}{\text{s}}$, respectively, when the forward speed is $\frac{1}{4}\frac{\text{m}}{\text{s}}$.

Measurement resolution: To obtain the angular wheel velocity, the frequency counting method (Petrella et al., 2007) was used, i.e. the encoder pulses inside a given sampling period were counted. The number of pulses per revolution (i.e. the number of windows in the code track of the encoders) was 378 and the sampling period was set to $\Delta t = 2.6$ s. Thus, according to Petrella et al. (2007) the measurement resolution was $\Delta\epsilon_r = \Delta\epsilon_l = \frac{2\pi}{378 \cdot 2.6} \approx 0.0064$.

Probability density functions: We obtained the distributions through experimental trials. Specifically, we used control inputs from $U_r \times U_l$ as the robot inputs and then measured the actual angular wheel velocities using the encoders. Since the output of the encoders was a pair of measured intervals $([\underline{w}_r, \overline{w}_r], [\underline{w}_l, \overline{w}_l])$, from each measurement we were able to determine the noise intervals, $[\underline{\epsilon}_r, \overline{\epsilon}_r]$ and $[\underline{\epsilon}_l, \overline{\epsilon}_l]$, of length $\Delta\epsilon_r$ and $\Delta\epsilon_l$, respectively, by using the fact that $([\underline{w}_r, \overline{w}_r], [\underline{w}_l, \overline{w}_l]) = ([u_r + \underline{\epsilon}_r, u_r + \overline{\epsilon}_r], [u_l + \underline{\epsilon}_l, u_l + \overline{\epsilon}_l])$ and the fact that $(u_r, u_l) \in \{U_r \times U_l\}$ was known. We obtained $\epsilon_i^{\min}$ ($\epsilon_i^{\max}$) by taking the minimum (maximum) over $\{\underline{\epsilon}_i^1, \ldots, \underline{\epsilon}_i^k\}$ ($\{\overline{\epsilon}_i^1, \ldots, \overline{\epsilon}_i^k\}$), where $[\underline{\epsilon}_i^j, \overline{\epsilon}_i^j]$, $j \in \{1, \ldots, k\}$, $i \in \{r, l\}$, was the noise interval, of length $\Delta\epsilon_i$, determined from the jth measurement of the encoder i, and k was the total number of measurements. Note that $n_i = \frac{|\epsilon_i^{\max} - \epsilon_i^{\min}|}{\Delta\epsilon_i}$, $i \in \{r, l\}$. Finally, the probabilities for equation (4) that defined the probability density functions, were equal to the number of times a particular noise interval was measured over k. For $k = 150$ (i.e. by using each control input from $\{U_r \times U_l\}$ 50 times) we obtained $-\epsilon_r^{\min} = \epsilon_r^{\max} = -\epsilon_l^{\min} = \epsilon_l^{\max} = 0.0096$ and the corresponding probabilities.

In this section we consider three case studies in which we (a)validateour main result (Theorem 1), (b) provide further insight into the complexity of the presented method, and (c) suggest how a potential runtime speed-up can be achieved.

9.1. *Case study 1*

In the first case study a BLTL formula and two different environments are considered. We use the simulation and experiment based satisfaction probabilities to verify Theorem 1.

The set of propositions was $\Pi = \{\pi_u, \pi_p, \pi_{t1}, \pi_{t2}, \pi_d\}$, where $\pi_u, \pi_p, \pi_{t1}, \pi_{t2}, \pi_d$ labeled the `unsafe`, `pick-up`, `test1`, `test2`, and the `drop-off` regions, respectively. The motion specification was:

Start from an initial state q_{init} *and reach a* `pick-up` *region within* 14 *time units and stay in it at least* 0.8 *time units, to pick-up the load. After entering the* `pick-up` *region, reach a* `test1` *region within* 5 *time units and stay in it at least* 1 *time units or reach a* `test2` *region within* 5 *time units and stay in it at least* 0.8 *time units. Finally, after entering the* `test1` *region or the* `test2` *region reach a* `drop-off` *region within* 4 *time units to drop off the load. Always avoid the* `unsafe` *regions.*

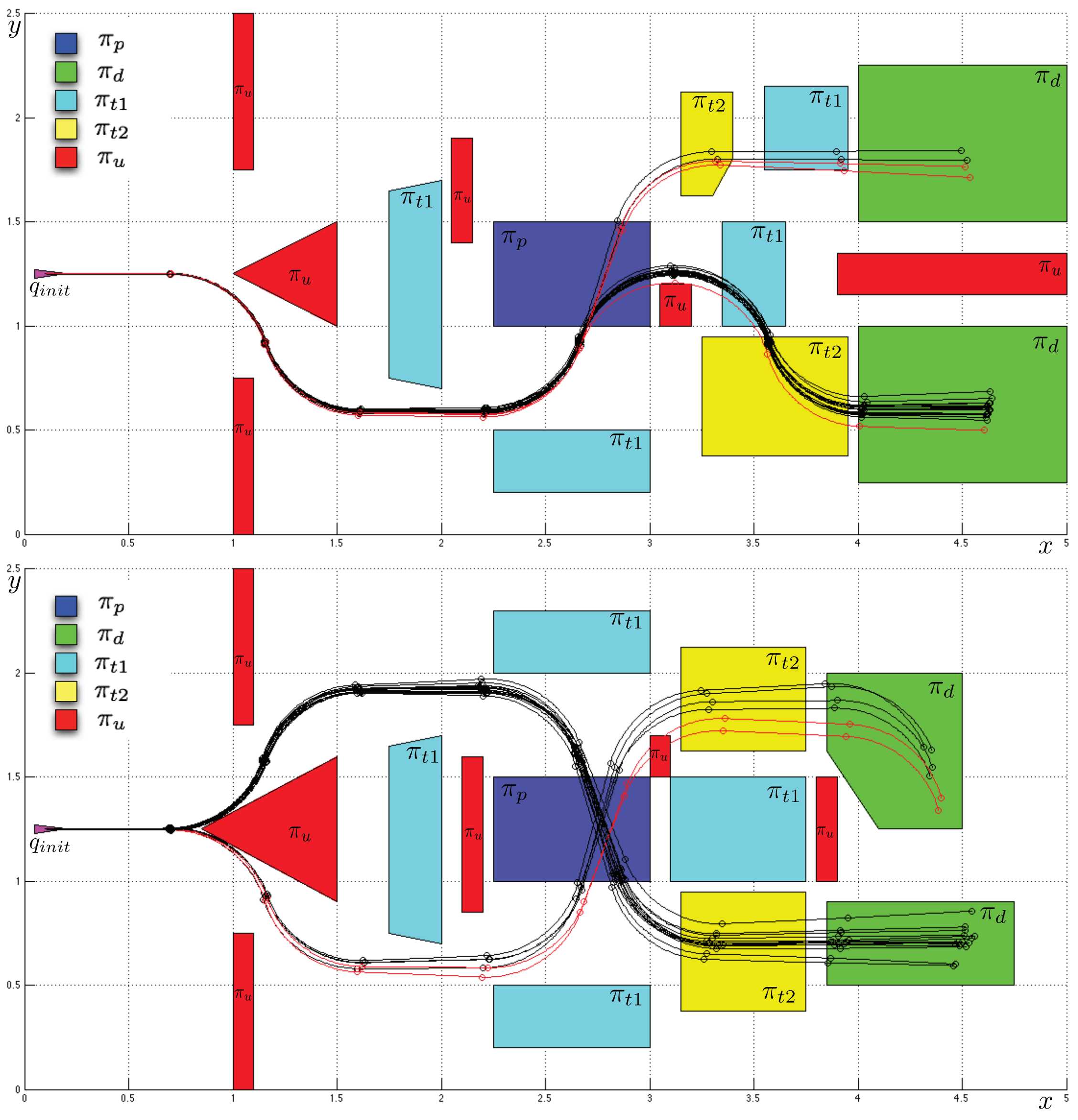


Fig. 7. 20 sample state (position) trajectories for cases A and B (to be read top to bottom). The `unsafe`, `pick-up`, `test1`, `test2`, and the `drop-off` regions are shown in red, blue, cyan, yellow, and green, respectively. Satisfying and violating trajectories are shown in black and red, respectively. Note that, in case A, the upper two red trajectories avoid the `unsafe` regions and visit the `pick-up`, `test2`, and the `drop-off` region in the correct order, but they violate the specification because they do not stay long enough in the `test2` region.

The specification translates to BLTL formula ϕ:

$$\phi = \neg\pi_u \mathbf{U}^{\leq 14}(\mathbf{G}^{\leq 0.8}\pi_p \wedge \neg\pi_u \mathbf{U}^{\leq 5} \\ ([\mathbf{G}^{\leq 1}\pi_{t1} \vee \mathbf{G}^{\leq 0.8}\pi_{t2}] \wedge \neg\pi_u \mathbf{U}^{\leq 4}\pi_d)) \qquad (10)$$

Two different environments are shown in Figure 7. The estimated probability $\hat{p}^*_M$ corresponding to environment A and B was 0.664 and 0.719, respectively. From equation (10) it followed that $K = 9$. The numerical values in the control synthesis procedure and the probability estimation procedure were as follows: $N = 10000$, $h = 0.6$, $g = 0.6$, $\delta = 0.05$, $c = 0.95$, $\alpha = \beta = 1$, and $e = 0.05$. For both environments, we found the vehicle control strategy through the method described in Section 8.

Since it is not possible to obtain the exact probability that the system given by equation (1), under the vehicle control strategy, generates a satisfying state trajectory, in order to verify our result (Theorem 8.1), we performed multiple runs of the BIE algorithm by simulating the system under the vehicle control strategy (using the same numerical values as stated above and by generating traces as described in Sections 6 and 7). We denote the resulting probability estimate as $\hat{p}_S$. Next, for each environment, we performed 50 experimental runs of the robot under the corresponding vehicle control strategy. A projector was used to display the environment and the state (position) trajectory was reconstructed using an OptiTrack

Table 1. Probability estimates of satisfying the specification.

Environment	$\hat{p}^*_M$	p_E	$\hat{p}_S$		
			Run 1	Run 2	Run 3
A	0.664	0.74	0.847	0.832	0.826
B	0.719	0.78	0.891	0.898	0.879

(http://www.naturalpoint.com/optitrack) system with eight cameras. In addition, the OptiTrack system was used to ensure that the robot always starts from the same initial state. We denote the resulting experiment based satisfaction probability (the number of satisfying runs over the total number of runs) as p_E.

In Figure 7 we show sample state trajectories and in Table 1 we compare the estimated probabilities obtained on the MDP, $\hat{p}^*_M$, with the experiment based satisfaction probabilities, p_E, and the estimated probabilities obtained by simulating the system, $\hat{p}_S$. The results support Theorem 8.1, since both p_E and $\hat{p}_S$ are bounded from below by $\hat{p}^*_M$. The discrepancy in the probabilities is mostly due to the conservative approximation of the uncertainty region in Section 6. Note that $\hat{p}_S$ is higher than p_E, i.e. the system given by equation (1), under the obtained vehicle control strategy, performed better than the physical robot, under the same vehicle control strategy. The reason for this is the fact that the system given by equation (1) assumes that the non-slipping and non-skidding conditions of the wheels are satisfied. However, during the experiments, slipping and skidding are present and, as an unmodeled noise, diminish the performance of the physical robot. In Figure 8 we show snapshots from the movie provided in Extension 1 which shows a sample experimental run of the robot in environment A.

Given $U_r \times U_l = \{(\frac{1+L}{4r}, \frac{1-L}{4r}), (\frac{1}{4r}, \frac{1}{4r}), (\frac{1-L}{4r}, \frac{1+L}{4r})\}$, $n_r = n_l = 3$, and $K = 9$, for both environments, the size of the corresponding MDPs was bounded above by $(3 \times 3 \times 3)^9$. Due to the size of the MDPs using the exact method to obtain a control policy was computationally infeasible. However, our method was able to produce a solution in approximately 2.2 hours and the actual number of states stored in the memory was approximately 3.5 million states ($<<(3 \times 3 \times 3)^9$).

9.2. Case study 2

In the second case study we considered a simple BLTL formula with an environment for which the exact MDP solution was computed. We compare the exact method with our method by investigating the computation times, the memory usage, the obtained satisfaction probabilities, and the obtained vehicle control strategies for both methods.

The set of propositions was $\Pi = \{\pi_u, \pi_p, \pi_{t1}\}$, where π_u, π_p, π_{t1} labeled the `unsafe`, `pick-up`, and the `test1` regions, respectively. The motion specification was:

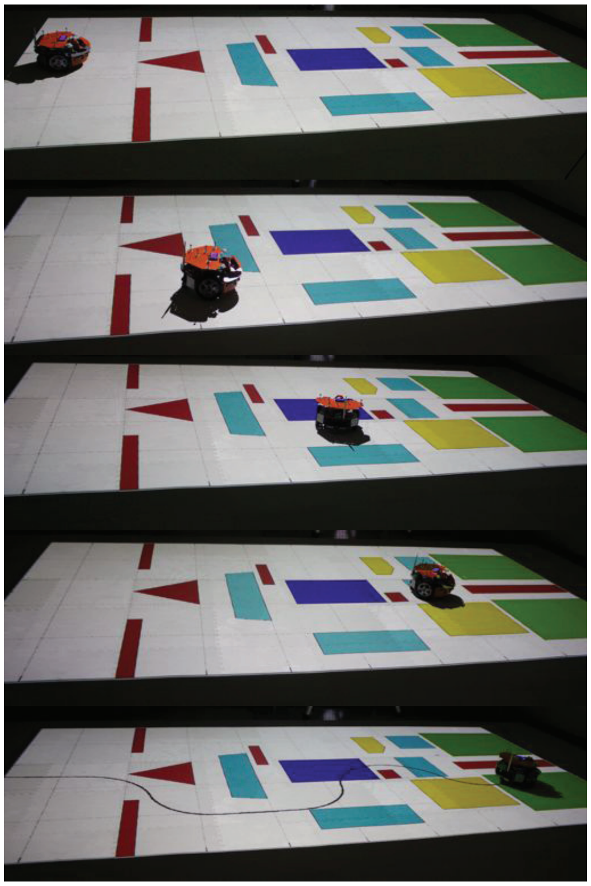


Fig. 8. Snapshots (to be read top to bottom) from a movie (see Extension 1) showing robot motion produced by applying the vehicle control strategy for environment A. The generated trajectory satisfied ϕ (equation (10)).

Start from an initial state q_{init} and reach a `pick-up` *region within* 6.5 *time units and stay in it at least* 1 *time unit, to pick-up the load. After entering the* `pick-up` *region, reach a* `test1` *region within* 5 *time units. Always avoid the* `unsafe` *regions.*

The specification translates to BLTL formula ϕ:

$$\phi = \neg\pi_u \mathbf{U}^{\leq 6.5}(\mathbf{G}^{\leq 1}\pi_p \wedge \neg\pi_u \mathbf{U}^{\leq 5}\pi_{t1}) \tag{11}$$

The numerical values were as given in Section 9.1. From equation (11) it followed that $K = 4$. The obtained estimated probability $\hat{p}^*_M$ was 0.945 and we found the vehicle control strategy through the method described in Section 8 in 6.6 min and the actual number of states stored in the memory was approximately 0.28 million states.

The size of the MDP was bounded from above by $(3 \times 3 \times 3)^4 \approx 0.5$ million states. For the particular MDP we were able to compute the exact control policy. To do so we only used a modified version of the control synthesis procedure of our algorithm as follows. In the control policy evaluation procedure we sampled all the paths through the MDP. This allowed us to obtain the exact probability that an MDP path crossing a state–action pair will generate a trace that

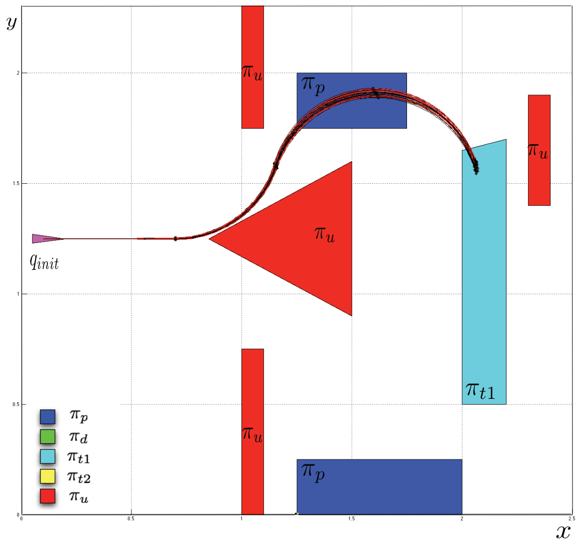


Fig. 9. 20 sample state (position) trajectories under the exact vehicle control strategy and under the vehicle control strategy returned by our method are shown in black and red, respectively. The `unsafe`, `pick-up`, and the `test1` regions are shown in red, blue, and cyan respectively. All state trajectories are satisfying.

satisfies ϕ. These probabilities were then used in the control policy improvement procedure, in which we updated the control policy μ by reinforcing the actions that led to the satisfaction of ϕ most often. The fact that each state in the MDP had exactly one incoming transition, i.e. the MDP had no cycles, is sufficient to see that the procedure state above resulted in the optimal control policy. Under the obtained optimal control policy the probability of satisfying the specification on the MDP was 1 and the optimal vehicle control strategy was obtained in 3 minutes.

In Figure 9 we show sample state trajectories obtained by simulating the system given by equation (1) under both vehicle control strategies. Note that the state trajectories perform the same and all of them were satisfying. By using our method only 0.28 million states were stored in the memory compared to 0.5 million states stored by the exact method. However, our method ran for 6.6 min compared to 3 min needed by the exact method. The latter is due to the fact that our method samples paths through the MDP and generates the corresponding traces in both the control synthesis and the probability estimation part of the algorithm. In particular, since the length of every path through the MDP was equal to $K = 4$, by using the exact method, the total number of paths through the MDP was $\frac{0.5\text{ million}}{4} = 0.125$ million, and for each path only one corresponding trace was generated. The number of sampled states, stored by our method, was 0.28 million, and therefore, only $\frac{0.28\text{ million}}{4} = 0.07$ million distinctive paths through the MDP were sampled. However, a total of 0.3 million traces was generated during the control synthesis and the probability estimation part of the algorithm.

9.3. *Case study 3*

For the case study presented in Section 9.1, the Matlab code executing the exact method (as proposed in Section 9.2) ran out of memory after approximately 6 h on a computer with a 2.5 GHz processor and a 16 GB RAM using a single sampling thread. On the same computer, our method produced the vehicle control strategy in approximately 2.2 h. Even though successful, the key limitation of the proposed approach is the computation time. However, the advantage of SMC is that sampling is highly parallelizable and a significant runtime speed-up can be achieved by increasing the number of sampling threads (for more information see the next section).

Additional runtime speed-up can be achieved by decreasing the confidence interval coefficient c or by increasing the half interval size δ, i.e. by trading-off statistical confidence for runtime efficiency. For example, for the case study presented in Section 9.1 corresponding to environment A, by decreasing c from 0.95 to 0.80, runtime was reduced to 0.68 h and approximately 2.5 million states were stored in the memory, i.e. the running time and the memory usage were reduced to approximately 30% and 70%, respectively, of their original values. In Figure 10 (case A when $c = 0.8$) we show sample state trajectories under the vehicle control strategy and in Table 2 we compare the estimated probabilities obtained on the MDP, $\hat{p}^*_M$, with the estimated probabilities obtained by simulating the system, $\hat{p}_S$, when $c = 0.8$. Note that, by reducing c, both $\hat{p}^*_M$ and $\hat{p}_S$ are reduced in comparison with the results presented in Table 1. This can also be seen in Figure 10 by noticing that the new vehicle control strategy performs worse, since less of the state space was explored by the algorithm.

Finally, we propose an approach that can achieve a runtime speed-up by using coarser incremental encoders. In particular, our algorithm can be initialized with a reduced resolution incremental encoders (by increasing $\Delta\epsilon_r$ and $\Delta\epsilon_l$), which reduces the size of the MDP. Then, if the user is not satisfied with the estimated satisfaction probability returned by the algorithm, one can incrementally increase the encoder resolution until a satisfactory vehicle control strategy is found or the original encoder resolution is reached. We tested this method by using a coarser version of the incremental encoders presented in Section 9, by setting the measurement resolution $\Delta\epsilon_r = \Delta\epsilon_l = 0.0096$, i.e. by setting $n_r = n_l = 2$. The size of the MDP was reduced from $(3 \times 3 \times 3)^9$ to $(3 \times 2 \times 2)^9$, i.e. it was reduced approximately 1500 times. For the case study presented in Section 9.1 corresponding to environment B, by decreasing $n_r = n_l = 3$ to $n_r = n_l = 2$, runtime was reduced to 1.1 h and approximately 2.7 million states were stored in the memory, i.e. the running time and the memory usage were reduced to approximately 50% and 77%, respectively, of their original values. Note that, by reducing the MDP size by a factor of 1500, we achieved 50% and 23% reductions in runtime and memory usage, respectively. From here, one can conclude that our algorithm scales gracefully with the

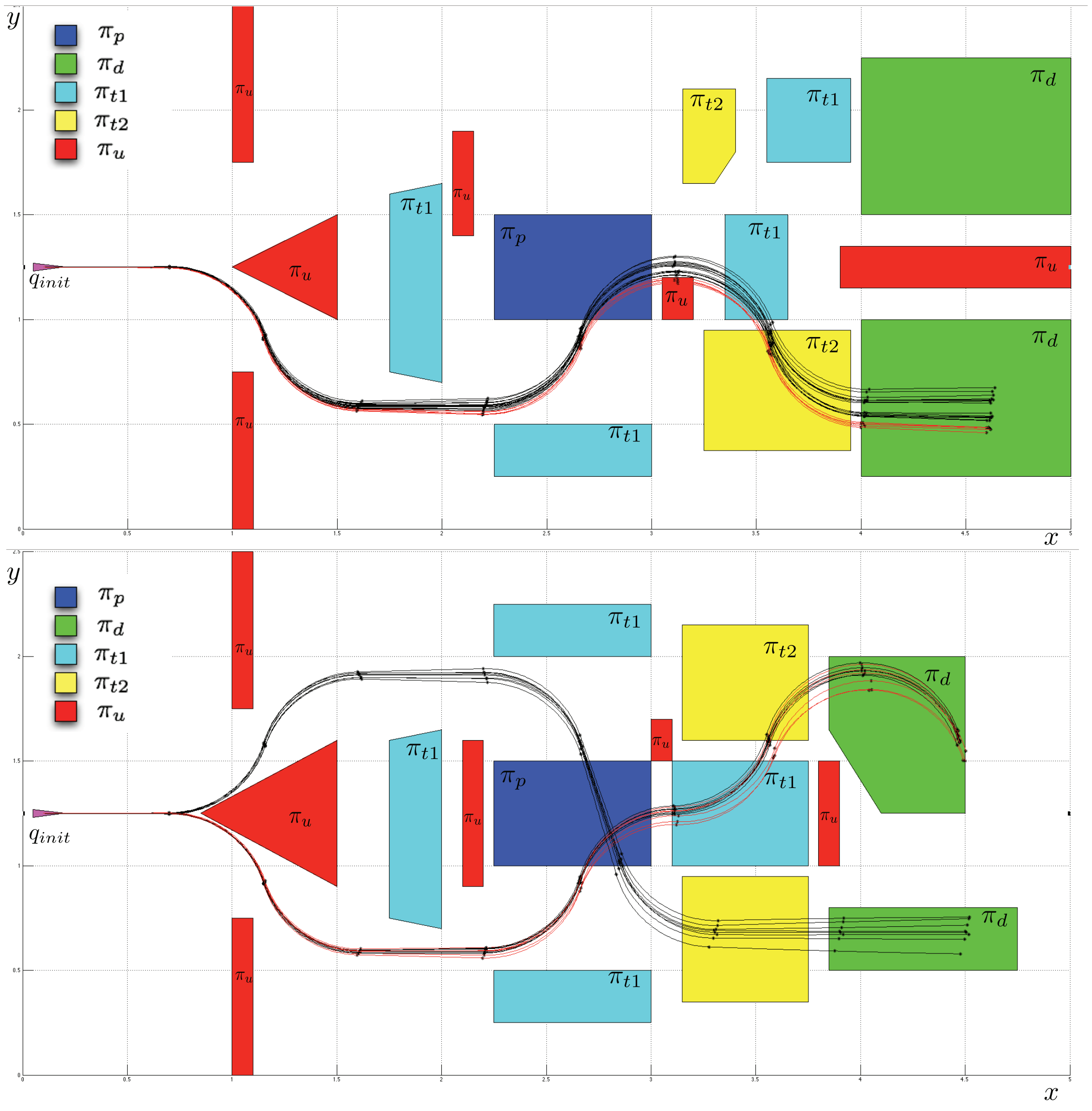


Fig. 10. 20 sample state (position) trajectories for cases A and B, when $c = 0.8$ and $n_r = n_l = 2$, respectively (to be read top to bottom). The `unsafe`, `pick-up`, `test1`, `test2`, and the `drop-off` regions are shown in red, blue, cyan, yellow, and green, respectively. Satisfying and violating trajectories are shown in black and red, respectively. Note that, in case B, the upper red trajectories avoid the `unsafe` regions and visit the `pick-up`, `test1`, `test2`, and the `drop-off` region in the correct order, but they violate the specification because they violate the time bounds.

size of the MDP. However, this is only the case when the MDP is a structured model (models that have some symmetry, e.g. robotics problem of motion planning) for which it has been shown that SMC for MDPs scales gracefully with the size of the MDP (for more details see Henriques et al. (2012)).

In Figure 10 (case B when $n_r = n_l = 2$) we show sample state trajectories under the new vehicle control strategy and in Table 2 we compare $\hat{p}^*_M$ and $\hat{p}_S$.

Table 2. Probability estimates of satisfying the specification with updated system parameters.

Environment	Updated Parameter	$\hat{p}^*_M$	$\hat{p}_S$		
			Run 1	Run 2	Run 3
A	$c = 0.8$	0.623	0.686	0.687	0.692
B	$n_r = n_l = 2$	0.684	0.793	0.801	0.796

10. Discussion and future work

We developed a feedback control strategy for a stochastic differential drive mobile robot such that the probability of satisfying a time constrained specification given in terms of a temporal logic statement is maximized. By mapping sensor measurements to an MDP we translated the problem to finding a control policy maximizing the probability of satisfying a BLTL formula on the MDP. The solution was based on SMC for MDPs and we showed that the probability that the vehicle satisfies the specification is bounded from below by the probability of satisfying the specification on the MDP.

The key limitation of the proposed approach is the computation time. The algorithm itself is extremely lightweight when compared to sampling. In particular, model checking and generation of control policies account for less than 10% of runtime. The remaining time is spent generating sample traces. There are two main reasons for this. First, when generating a trace corresponding to an uncertainty region, a computationally expensive step of taking the intersection between the uncertainty region and all of the regions of interest is performed (see Definition 7.1, Section 7). Second, note that the system is continuous both in space and time. Thus, at each time step, when constructing an uncertainty region, the algorithm performs multiple integrations of the system.

Faster sampling methods have the potential to significantly decrease the runtime. One major advantage of using SMC is that sampling is highly parallelizable and significant gains can be obtained by increasing the number of sampling threads. In particular, results from Henriques et al. (2012) show that by having 10 threads, runtime can be reduced to under 25% of its original value. To address the fact that sampling (i.e. generating sample traces) accounts for the majority of the runtime, future work includes improving the sampling performance and making the implementation fully parallel.

Another direction for our future research is to investigate how increasing the control rate or having more available controls will affect the vehicle control quality and the computation time. It is obvious that this will increase the size of the MDP which, due to the fact that sampling accounts for more than 90% of runtime, will increase the computation time. Also it is easy to see that having more measurements and controls will result in an improved vehicle control strategy. However, we believe that this is an important issue that needs to be treated separately. To address the problem of discrepancy between the probabilities obtained on the MDP and the probabilities obtained by simulating the system, we also plan to use a less conservative uncertainty model. In particular, we plan to implement the approach presented in Althoff and Dolan (2011) where, given the mathematical model of a vehicle and a bound on uncertainty, the authors construct a tight overapproximated reachability set for a finite lookahead horizon. Future work also includes experimental implementations in which the algorithm from this paper is used to control a ground robot that is simultaneously deployed with an autonomous aircraft that can observe the environment, formulate the time-bounded temporal logic specifications for the ground vehicle, and provide position information to restart the control algorithm for the ground vehicle.

Acknowledgements

Preliminary results from this paper were presented at the 2013 IEEE international conference on robotics and automation (ICRA 2013). The authors gratefully acknowledge David Henriques from Carnegie Mellon University for comments on an earlier draft of this work. Also, the authors would like to thank Benjamin Troxler, Michael Marrazzo, and Matt Buckley from Boston University for their help with the experiments.

Funding

This work was supported by the ONR MURI (grant number N00014-10-10952) and by the NSF (grant number CNS-0834260).

Notes

1. Throughout the paper, we relax the notion of partition by allowing the endpoints of the intervals to overlap.
2. Since the regions of interest are non-overlapping it follows that $o_i \in \Pi \cup \emptyset$.

References

Abate A, D'Innocenzo A and Di Benedetto M (2011) Approximate abstractions of stochastic hybrid systems. *IEEE Transactions on Automatic Control* 56(11): 2688–2694.

Alterovitz R, Simeon T and Goldberg K (2007) The stochastic motion roadmap: a sampling framework for planning with Markov motion uncertainty. In: *Proceedings of robotics: science and systems III (RSS 2007)*, Atlanta, GA, USA, 27–30 June 2007, pp. 233–241.

Althoff M and Dolan J (2011) Set-based computation of vehicle behaviors for the online verification of autonomous vehicles. In: *Proceedings of 14th International IEEE conference on intelligent transportation systems (ITSC)*, Washington DC, USA, 5–7 October 2011, pp. 1162–1167.

Bai H, Hsu D and Lee WS (2013) Planning how to learn. In: *Proceedings of International conference on robotics and automation (ICRA)*, Karlsruhe, Germany, 6–10 May 2013, pp. 2853–2859.

Baier C and Katoen JP (2008) *Principles of Model Checking*. Cambridge, MA: MIT Press.

Balkcom D and Mason MT (2000) Time optimal trajectories for bounded velocity differential drive robots. In: *Proceedings of IEEE International conference on robotics and automation (ICRA)*, Volume 3, San Francisco, CA, USA, 24–28 April 2000, pp. 2499–2504.

Bhatia A, Kavraki L and Vardi M (2010) Motion planning with hybrid dynamics and temporal goals. In: *Proceedings of 49th IEEE conference on decision and control (CDC)*, Atlanta, GA, USA, 15–17 December 2010, pp. 1108–1115.

Bhatia A, Maly MR, Kavraki LE and Vardi MY (2011) Motion planning with complex goals. *IEEE Robotics Automation Magazine* 18(3): 55–64.

Cizelj I and Belta C (2012) Probabilistically safe control of noisy dubins vehicles. In: *Proceedings of IEEE intelligent robots and systems (IROS) conference*, Algarve, Portugal, 7–12 October 2012, pp. 2857–2862.

Cizelj I and Belta C (2013) Control of noisy differential-drive vehicles from time-bounded temporal logic specifications. In:

Proceedings of IEEE International conference on robotics and automation (ICRA), Karlsruhe, Germany, 6–10 May 2013, pp. 2021–2026.

Clarke E, Grumberg O and Peled DA (1999) *Model Checking*. Cambridge, MA: MIT Press.

Ding XC, Lazar M and Belta C (2012) Receding horizon temporal logic control for finite deterministic systems. In: *Proceedings of IEEE American control conference (ACC)*, Montreal, QC, Canada, 27–29 June 2012, pp. 715–720.

D'Innocenzo A, Abate A, Benedetto MDD, et al. (2008) Approximate abstractions of discrete-time controlled stochastic hybrid systems. In: *Proceedings of IEEE conference on decision and control (CDC)*, Cancun, Mexico, 9–11 December 2008, pp. 221–226.

Ekekwe N, Etienne-Cummings R and Kazanzides P (2007) Incremental encoder based position and velocity measurements VLSI chip with serial peripheral interface. In: *Proceedings of IEEE International symposium on circuits and systems (ISCAS) 2007*, New Orleans, LA, USA, 27–30 May 2007, pp. 3558–3561.

Fainekos G, Girard A, Kress-Gazit H, et al. (2009) Temporal logic motion planning for dynamic robots. *Automatica* 45(2): 343–352.

Fraichard T and Mermond R (1998) Path planning with uncertainty for car-like robots. In: *Proceedings of IEEE International conference on robotics and automation (ICRA)*, Leuven, Belgium, 16–20 May 1998, pp. 27–32.

Girard A (2007) Approximately bisimilar finite abstractions of stable linear systems. In: *Proceedings of International conference on hybrid systems: Computation and control (HSCC)*, Pisa, Italy, 3–5 April 2007, pp. 231–244.

Grady DK, Moll M and Kavraki LE (2013) Combining a POMDP abstraction with replanning to solve complex, position-dependent sensing tasks. Arlington, VA: AAAI Press.

Henriques D, Martins J, Zuliani P, et al. (2012) Statistical model checking for Markov decision processes. In: *Proceedings of 9th International conference on quantitative evaluation of systems (QEST)*, London, UK, 17–20 September 2012, pp. 84–93.

Jha SK, Clarke EM, Langmead CJ, et al. (2009) A Bayesian approach to model checking biological systems. In: *Proceedings of International conference on computational methods in systems biology (CMSB)*, Bologna, Italy, 31 August–1 September 2009, pp. 218–234.

Julius AA and Pappas GJ (2009) Approximations of stochastic hybrid systems. *IEEE Transactions on Automatic Control* 54(6): 1193–1203.

Karaman S and Frazzoli E (2008) Vehicle routing problem with metric temporal logic specifications. In: *Proceedings of IEEE conference on decision and control (CDC)*, Cancun, Mexico, 9–11 December 2008, pp. 3953–3958.

Kloetzer M and Belta C (2008a) Dealing with non-determinism in symbolic control. In: *Hybrid Systems: Computation and Control: 11th International Workshop* (Lecture Notes in Computer Science). Berlin; Heidelberg, Germany: Springer, pp. 287–300.

Kloetzer M and Belta C (2008b) A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transactions on Automatic Control* 53(1): 287–297.

Koymans R (1990) Specifying real-time properties with metric temporal logic. *Real-Time Systems* 2(4): 255–299.

Kress-Gazit H, Fainekos GE and Pappas GJ (2007) Where's Waldo? Sensor-based temporal logic motion planning. In: *Proceedings of IEEE International conference on robotics and automation (ICRA) 2007*, Shanghai, 9–13 May 2011, pp. 3116–3121.

Lahijanian M, Andersson SB and Belta C (2012) Temporal logic motion planning and control with probabilistic satisfaction guarantees. *IEEE Transactions on Robotics* 28(2): 396–409.

LaValle SM (2006) *Planning Algorithms*. Cambridge: Cambridge University Press.

Loizou SG and Kyriakopoulos KJ (2004) Automatic Synthesis of multi-agent motion tasks based on LTL specifications. In: *Proceedings of 43rd IEEE conference on decision and control (CDC)*, Volume 1, Atlantis, Bahamas, 14–17 December 2004, pp. 153–158.

Melchior N and RSimmons (2007) Particle RRT for path planning with uncertainty. In: *Proceedings of IEEE International conference on robotics and automation (ICRA)*, Roma, Italy, 10–14 April 2007, pp. 1617–1624.

Petrella R, Tursini M, Peretti L, et al. (2007) Speed measurement algorithms for low-resolution incremental encoder equipped drives: A comparative analysis. In: *Proceedings of International Aegean conference on electrical machines and power electronics (ACEMP)*, Bodrum, Turkey, 10–12 September 2007, pp. 780–787.

Shkolnik A, Walter M and Tedrake R (2009) Reachability-guided sampling for planning under differential constraints. In: *Proceedings of IEEE International conference on robotics and automation (ICRA)*, Kobe, Japan, 12–17 May 2009, pp. 4387–4393.

Tabuada P and Pappas GJ (2006) Linear time logic control of discrete-time linear systems. *IEEE Transactions on Automatic Control* 51(12): 1862–1877.

Wongpiromsarn T, Topcu U and Murray RM (2009) Receding horizon temporal logic planning for dynamical systems. In: *48th IEEE conference on decision and control (CDC)*, Shanghai, 16–18 December 2009, pp. 5997–6004.

Yordanov B, Tumova J, Cerna I, et al. (2012) Temporal Logic Control of discrete-time piecewise affine systems. *IEEE Transactions on Automatic Control* 57(6): 1491–1504.

Zuliani P, Platzer A and Clarke EM (2010) Bayesian statistical model checking with application to simulink/stateflow verification. In: *Proceedings of International conference on hybrid systems: computation and control (HSCC)*, Stockholm, Sweden, 12–16 April 2010, pp. 243–252.

Appendix: Index to Multimedia Extensions

The multimedia extension page is found at http://www.ijrr.org

Table of Multimedia Extensions

Extension	Media type	Description
1	Video	Robot implementation in environment *A*