Integrated Motion Planning and Control Under Metric Interval Temporal Logic Specifications

Fernando S. Barbosa[1], Lars Lindemann[2], Dimos V. Dimarogonas[2] and Jana Tumova[1]

Abstract— **This paper proposes an approach that combines motion planning and hybrid feedback control design in order to find and follow trajectories fulfilling a given complex mission involving time constraints. We use Metric Interval Temporal Logic (MITL) as a rich and rigorous formalism to specify such missions. The solution builds on three main steps: (i) using sampling-based motion planning methods and the untimed version of the mission specification in the form of Zone automaton, we find a sequence of waypoints in the workspace; (ii) based on the clock zones from the satisfying run on the Zone automaton, we compute time-stamps at which these waypoints should be reached; and (iii) to control the system to connect two waypoints in the desired time, we design a low-level feedback controller leveraging Time-varying Control Barrier Functions. Illustrative simulation results are included.**

I. Introduction

In recent years, different variants of temporal logic specifications have been established to complement the traditional A-to-B motion planning algorithms with more complex, structured missions. For instance, with the use of Linear Temporal Logic (LTL), one can formalize properties such as “Visit region A, then B, while avoiding a dangerous area C”. In order to allow for time constraints in the mission specification, such as “Visit A within 10 to 20 time units, then visit B no earlier than 30 time units after reaching A, while avoiding C”, a timed temporal logic is required, such as Metric Interval Temporal Logic (MITL). A general approach to planning under temporal logic tasks builds on finding a suitable discrete abstraction of the system dynamics, such as a finite bisimulation, or a graph obtained from a sampling-based motion planning method, ensuring that a transition in the abstraction can be followed via an application of a certain control law [1], [2]. When it comes to planning under timed temporal logic tasks, finding such a discrete abstraction becomes much more challenging. Related literature often assumes that a discrete system model is given, e.g., in a form of a Weighted Timed Automaton [3], [4]. Another approaches use time-sampling to obtain a discretization [5], or work directly with the original system dynamics and use a restricted fragment of Signal Temporal Logic (STL) for control over a short time horizon [6].

This work proposes an approach to find and follow a trajectory fulfilling a time-bounded MITL task that integrates sampling-based motion planning with low-level feedback controller design. A general overview is presented in Fig. 1. A given MITL task specification is first translated into a Timed Automaton (TA) and then to a Zone Automaton (ZA), which is its time-abstract representation. The ZA is used by an RRT*-based algorithm to find an obstacle-free path – a sequence of waypoints – in the workspace enclosed in a sequence of polytopes. Times to reach these waypoints are calculated by using clock zones of the ZA as constraints of a Linear Program (LP). To execute the timed path, we propose the use of the quadratic program-based Time-varying Control Barrier Functions controller derived in [6]. Such a controller ensures that if the system is within a convex, obstacle-free space, such as a polytope computed by the sampling-based planner, it is able to navigate from an initial to a goal configuration within a given time window.

This work was supported in part by SRA ICT TNG Project STaRT, in part by EU H2020 Co4Robots project, in part by VR (Swedish Research Council) and in part by Knut och Alice Wallenberg Foundation (KAW).

[1]Division of Robotics, Perception and Learning (RPL), KTH Royal Institute of Technology, Stockholm, Sweden. {fdsb,tumova}@kth.se

[2]Division of Decision and Control Systems, KTH Royal Institute of Technology, Stockholm, Sweden. {llindem, dimos}@kth.se

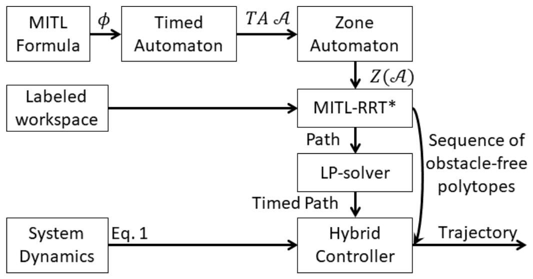


Fig. 1. The approach.

The works most closely related to ours include sampling-based motion planning under different temporal logic specifications. Rapidly-exploring Random Tree (RRT) technique was used to find a motion plan that fulfills μ-calculus specifications [7], while RRT* [8] was used for minimum-violation motion planning under finite-LTL specifications [9], and later on complemented to deal with syntactically co-safe LTL (sc-LTL) in workspaces with limited perception horizons [10]. One of the recent works proposes an RRT*-based algorithm to find a motion plan that maximally-satisfies a Signal Temporal Logic (STL) formula [5]. The method relies on time-sampling and uses quantitative semantics provided by the STL to guide both sampling and steering towards the most robust solution. A reactive motion planner that takes into account imperfect state information and uses feedback-based information roadmaps (FIRMs) to maximize the probability of satisfying a high level specification given in LTL has been proposed in [11]. Lastly, [12] proposes a framework capable of planning in dynamically changing environments with dynamic, local MTL mission requirements.

Other closely related work includes literature on control barrier functions, which have been used in [13], [14], [15], [16] to establish forward invariance of certain sets, defining safety specifications for dynamical systems. Additionally, [14], [15], [16] propose a quadratic program that guarantees safety while aiming for reachability specifications. However, it has been shown that the system may not reach the goal configuration, although remaining safe; if it is reached, it is not known in advance how long it takes to accomplish the task. Our previous work [6] proposes to use time-varying control barrier function, opposed to static ones as in the aforementioned works, to satisfy a subclass of Signal Temporal Logic tasks.

To our best knowledge, none of the state-of-the-art works addresses the problem of sampling-based motion planning for a nonlinear dynamical system in a complex workspace under timed temporal logic specification with closed-loop control guarantees and without the necessity of introducing time sampling.

The remainder of the paper is structured as follows. Sec. II describes preliminary theory. We formally define our problem in Sec. III together with the proposed approach, which is detailed in three sections: motion planning algorithm (Sec. IV), timed path calculation (Sec. V), and low-level controller (Sec. VI). Lastly, the approach is analyzed in Sec. VII and case studies are presented in Sec. VIII.

II. Preliminaries

Real numbers are denoted by $\mathbb{R}$, while $\mathbb{R}^n$ is the n-dimensional real vector space. $\mathbb{N}$ denotes nonnegative integers. Time is denoted by $\mathbb{T} = \mathbb{R}_{\geq 0}$, the set of nonnegative real numbers, and the set of Booleans by $\mathbb{B}$.

A convex polytope in $\mathbb{R}^n$ can be represented by its H-representation, defined as an intersection of a finite number of half-spaces, written in the form of a matrix inequality as $Ax \leq b$, $x \in \mathbb{R}^n$.

A. System Dynamics

Let $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{u} \in \mathbb{R}^m$, $\boldsymbol{d} \in \mathfrak{D} \subset \mathbb{R}^n$, where $\mathfrak{D} := \{\boldsymbol{d} \in \mathbb{R}^n | \|\boldsymbol{d}\| \leq D\}$ for some $D \geq 0$, be the state, input, and unknown disturbance, respectively, of a nonlinear system

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + g(\boldsymbol{x})\boldsymbol{u} + \boldsymbol{d}(t), \tag{1}$$

with locally Lipschitz continuous functions $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ such that $g(\boldsymbol{x})g(\boldsymbol{x})^T$ is positive definite for all $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{d} : \mathbb{R}_{\geq 0} \to \mathfrak{D}$ is piecewise continuous.

The bounded subset $W \subset \mathbb{R}^n$ defines a workspace, divided into *obstacles* $W_{obs} \subset W$ and *free space* $W_{free} = W \setminus W_{obs}$.

The free space W_{free} is divided into k mutually disjunct open regions of interest, $\mathcal{W}_{free} = \{W^1_{free}, \ldots, W^k_{free}\}$, with the property that the union of their closures is W_{free}. Each state $x \in \mathbb{R}^n$ is associated with a subset of atomic propositions AP through a labeling function $L : W \to 2^{\mathsf{AP}}$. Without loss of generality, we assume that each region of interest W^i_{free} satisfies that $L(x) = L(x')$, for all $x, x' \in W^i_{free}$ and hence, with a slight abuse of notation, we use $L : \mathcal{W}_{free} \to 2^{\mathsf{AP}}$ as a labeling function of regions.

Definition 1 (Timed path)**.** *A finite trajectory* $\mathbf{x}$ *of the system (1) in* W_{free} *defines a timed path* $(\mathbf{p}, \mathbf{t}) = (x_0, t_0)(x_1, t_1) \ldots (x_\ell, t_\ell)$*, such that for all* $i \in \{0, \ldots \ell\}$*:*

- $\mathbf{p} = x_0 x_1 \ldots x_\ell$*, with* $x_i = \mathbf{x}(t_i)$*,*
- $\mathbf{t} = t_0 t_1 \ldots t_\ell$ *is a sequence of* time-stamps*, with* $t_i \in \mathbb{T}$*,* $t_0 = 0$*,* $t_i \geq t_{i-1}$*, ;*
- $L(x_i)$ *holds true during the time interval* (t_i, t_{i+1})*.*

Definition 2 (Timed word)**.** *A trajectory* $\mathbf{x}$ *with a timed path* $(\mathbf{p}, \mathbf{t}) = (x_0, t_0)(x_1, t_1) \ldots (x_\ell, t_\ell)$ *generates a* timed word $w(\mathbf{x}) = w(\mathbf{p}, \mathbf{t}) = (L(x_0), t_0)(L(x_1), t_1) \ldots (L(x_\ell), t_\ell)$*.*

Given a timed word $w = (L(x_0), t_0) \ldots (L(x_\ell), t_\ell)$*, let* $w^t = (L(x_i), t_i - t)(L(x_{i+1}), t_{i+1} - t) \ldots (L(x_\ell), t_\ell - t)$ *be a* suffix *of that word starting at time t, where* $t_{i-1} < t \leq t_i$*.*

B. Time-bounded MITL

A *time-bounded Metric Interval Temporal Logic* (MITL) formula over a set of atomic propositions (AP) and time intervals of the form $[a, b]$, $a, b \in \mathbb{N}$ and $a < b$ is defined as:

$$\phi := \top \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \mathcal{U}_{[a,b]} \phi_2 \mid \mathcal{F}_{[a,b]}\phi \mid \mathcal{G}_{[a,b]}\phi, \tag{2}$$

where the proposition $p \in \mathsf{AP}$ and $\neg$, $\wedge$, are standard negation and conjunction, and $\mathcal{U}$, $\mathcal{F}$ and $\mathcal{G}$ correspond to temporal operators until, eventually and always, respectively.

Definition 3 (MITL [17])**.** *The satisfaction of a time-bounded MITL formula* ϕ *over* AP *on a timed word* $w = (w_0, t_0) \ldots (w_\ell, t_\ell)$ *is defined as follows:*

$$\begin{aligned}
w \models p &\iff p \in w_0 \\
w \models \neg\phi &\iff w \not\models \phi \\
w \models \phi_1 \wedge \phi_2 &\iff w \models \phi_1 \textit{ and } w \models \phi_2 \\
w \models \phi_1 \mathcal{U}_{[a,b]} \phi_2 &\iff \exists t'. t' \in [a, b] \textit{ s.t. } w^{t'} \models \phi_2 \\
&\qquad \textit{and } \forall t'' < t', w^{t''} \models \phi_1 \\
w \models \mathcal{F}_{[a,b]}\phi &\iff \exists t'. t' \in [a, b] s.t. w^{t'} \models \phi \\
w \models \mathcal{G}_{[a,b]}\phi &\iff \forall t'. t' \in [a, b] \Rightarrow w^{t'} \models \phi.
\end{aligned}$$

C. Timed and Zone Automata

Let $C = \{c_1, \ldots, c_{|C|}\}$ be a finite set of clocks. A clock valuation is a function $\nu : C \to \mathbb{T}$ that assigns a value to each clock. $\nu + \delta$ maps every clock c to the value $\nu(c) + \delta$, for $\delta \in \mathbb{R}$. For $R \subseteq C$, $\nu[R := 0]$ assigns 0 to each $c \in R$, and agrees with ν over the rest of the clocks. A *clock constraint* g is defined as

$$g := \top \mid \neg g \mid c \bowtie \psi \mid g_1 \wedge g_2$$

where $c \in C$, $\psi \in \mathbb{N}$ and $\bowtie \in \{<, >, \leq, \geq, =\}$. The set of clock constraints over C is $\Phi(C)$.

Similarly, a *clock zone* φ is defined as

$$\varphi := c \bowtie \psi \mid (c_1 - c_2) \bowtie \psi \mid \varphi_1 \wedge \varphi_2$$

where $c, c_1, c_2 \in C$. $\varphi \Uparrow$ denotes the set of valuations $\nu + \delta$ for $\nu \in \varphi$ and $\delta \in \mathbb{T}$. For $R \subseteq C$, $\varphi[R := 0]$ denotes the set of clock valuations $\nu[R := 0]$, where ν satisfies φ. The set of clock zones over C is $\Gamma(C)$.

Definition 4 (Timed Automaton [18])**.** *A Timed Automaton (TA)* $\mathcal{A}$ *is a tuple* $\mathcal{A} = (Q, Q_0, C, Inv, E, F, \mathsf{AP})$ *where*

Q is a finite set of locations; $Q_0 \subseteq Q$ is the set of initial locations; C is a finite set of clocks; $Inv : Q \to \Phi(C)$ is the invariant function; $E \subseteq Q \times Q \times 2^{\mathsf{AP}} \times \Phi(C) \times 2^C$ is the set of edges; $F \subseteq Q$ is the set of accepting locations; and AP is a finite set of atomic propositions.

A state (q, ν) of a TA is given by location $q \in Q$, and clock valuation ν. A TA has two types of transitions:

- time passing: $(q, \nu) \xrightarrow{\delta} (q, \nu + \delta)$ if ν satisfies $Inv(q)$ and $(\nu + \delta)$ satisfies $Inv(q)$, where $\delta \in \mathbb{T}$;
- discrete jump: $(q, \nu) \xrightarrow{w} (q', \nu')$ if $(q, q', w, g, R) \in E$, ν satisfies g, $\nu' = \nu[R := 0]$ and ν' satisfies $Inv(q')$.

A run r of a TA $\mathcal{A}$ with initial state (q_0, ν_0) over a timed word $w = (w_0, t_0) \dots (w_\ell, t_\ell)$ is a finite sequence of transitions $(q_0, \nu_0) \xrightarrow{\delta_1}\xrightarrow{w_1} (q', \nu') \xrightarrow{\delta_2}\xrightarrow{w_2} \dots (q'', \nu'')$ such that $t_0 = 0$ and $t_i = t_{i-1} + \delta_i$, $\forall i \geq 1$. A timed word w is accepted by TA $\mathcal{A}$ if there exists an accepting run r of $\mathcal{A}$ over w such that the last location of its last state is accepting. The language $\mathcal{L}(\mathcal{A})$ of $\mathcal{A}$ is the set of all accepted words by $\mathcal{A}$. Its untimed language $\mathcal{L}_{\mathrm{un}}(\mathcal{A})$ is the set of all untimed words accepted by $\mathcal{A}$, which are the projections of the accepted words of $\mathcal{A}$ onto their first components. Untimed words are thus sequences over 2^{AP}.

Lemma 1 ([17])**.** *Any MITL formula ϕ over AP can be algorithmically translated into a TA $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A})$ is exactly the set of timed words that satisfy ϕ.*

Definition 5 (Zone Automaton)**.** *A Zone automaton $Z(\mathcal{A})$ is a tuple $Z(\mathcal{A}) = (Z, Z_0, E_z, F_z, \mathsf{AP})$, where $Z = Q \times \Gamma(C)$ is the set of states, Q is a set of locations, C is a set of clocks; $Z_0 \subseteq Z$ is set of initial states; $E_Z : Z \times 2^{\mathsf{AP}} \to Z$ is the transition function; $F_z \subseteq Z$ is the set of accepting states, and AP is the alphabet, i.e. a finite set of atomic propositions.*

A Zone automaton is in fact a finite automaton and as such, it defines a language $\mathcal{L}(Z(\mathcal{A}))$ of words over 2^{AP} that are accepted by its finite accepting runs.

Lemma 2 ([18])**.** *Any TA $\mathcal{A}$ can be algorithmically translated into a ZA $Z(\mathcal{A})$, such that $\mathcal{L}(Z(\mathcal{A})) = \mathcal{L}_{\mathrm{un}}(\mathcal{A})$.*

III. Problem Definition and Approach

Our goal is to design a control law for a dynamical system that produces a trajectory satisfying a complex task in a complex environment involving time bounds. While standard control theory methods are not immediately suited for handling high-level, long-term tasks, motion planning algorithms are, but require certain – often strong – assumptions on the low-level controllers. This paper bridges the two approaches; we allow for handling high-level tasks via integrated motion planning and control scheme. In addition to a sequence of waypoints (timed path) in the workspace that the system should go through, our motion planning algorithm generates properties that need to be met by its low-level controller. We propose such a controller and this way, we not only obtain a satisfying timed path, but also an actual satisfying trajectory of the original system. We state our problem as follows:

Problem 1. *Given a system (1), a labeled workspace W partitioned into W_{obs} and W_{free}, and a mission specification ϕ in time-bounded MITL, find a control law $\mathbf{u}$ producing a trajectory $\mathbf{x}$ that generates a timed word $w(\mathbf{x})$ satisfying ϕ.*

Our approach to Problem 1 is illustrated in Fig. 1. First, the time-bounded MITL mission specification ϕ is translated into a Timed Automaton $\mathcal{A}$ (Lemma 1) and thereafter to its time-abstract representation Zone automaton $Z(\mathcal{A})$ (Lemma 2). The ZA is used in a sampling-based motion planning algorithm, named MITL-RRT*. It returns (i) a sequence of waypoints that give a path satisfying an untimed version of the specification, and (ii) assumptions to be met by the low-level controller in the form of a sequence of obstacle-free polytopes within which the trajectory of the system has to stay.

Second, given a sequence of waypoints, we find appropriate time stamps so that the corresponding timed path satisfies the specification ϕ. To that end, we exploit the structure of the ZA and use its clock zones as constraints of a Linear Program (LP).

Third, we design a hybrid feedback control law that, applied to system (1), tracks the timed path returned by the motion planner while staying within the obstacle-free polytopes returned by MITL-RRT*.

This approach can be formalized by decomposing Problem 1 into the following three sub-problems, addressed in Sec. IV, V, and VI, respectively:

Problem 2. *Given a workspace W and a mission specification ϕ in bounded MITL translated into an equivalent Timed Automaton $\mathcal{A}$, and into an untimed Zone Automaton $Z(\mathcal{A})$, find a path $\mathbf{p} = x_0 x_1 \dots x_\ell$ and a sequence of convex polytopes $\boldsymbol{\pi} = \pi_1 \pi_2 \dots \pi_\ell$ such that:*

- *every $\pi_i \in \boldsymbol{\pi}$ is convex and obstacle-free;*
- *the i-th polytope π_i contains x_{i-1} and x_i;*
- *$\mathbf{p}$ is accepted by $Z(\mathcal{A})$, i.e. there exists a timed path $(\mathbf{p}, \mathbf{t}) = (x_0, t_0)(x_1, t_1) \dots (x_\ell, t_\ell)$ such that the timed word $w(\mathbf{p}, \mathbf{t})$ satisfies ϕ.*

Problem 3. *Given a path $\mathbf{p} = x_0 x_1 \dots x_\ell$ and a specification ϕ in bounded MITL, find a sequence of time stamps $\mathbf{t} = t_0 t_1 \dots t_\ell$ that, together with $\mathbf{p}$, generates a timed path $(\mathbf{p}, \mathbf{t}) = (x_0, t_0)(x_1, t_1) \dots (x_\ell, t_\ell)$ such that the timed word $w(\mathbf{p}, \mathbf{t})$ satisfies ϕ.*

Problem 4. *Design a control law $\boldsymbol{u}(\boldsymbol{x}, t)$ capable of driving a system (1) throughout a timed path $(\mathbf{p}, \mathbf{t})$ while remaining inside a sequence $\boldsymbol{\pi}$ of obstacle-free, convex polytopes.*

IV. Motion Planning

To solve Problem 2, suppose that the MITL formula ϕ has been translated to a Timed Automaton $\mathcal{A}$ and to its time-abstract Zone automaton $Z(\mathcal{A}) = (Z, Z_0, E_z, F_z, \mathsf{AP})$ using an existing algorithm, such as [19], [20].

Our approach builds on sampling-based motion planning algorithms, in particular on RRT*, which was originally designed for the anytime, incremental, asymptotically optimal motion planning while avoiding obstacles. Loosely speaking, RRT* incrementally builds a tree $G = (V, E)$ whose nodes V are states of the state-space and an edge $e \in E$ connects two nodes only if there exists a controller capable of taking the system from the parent to the child node. RRT* iteratively draws a sample from W and attempts to connect it to a near node that already is part of the tree. Namely, given a cost function J, the algorithm connects the sample to the optimal parent node, which ensures the lowest cost to get to it from the root and whose trajectory is obstacle-free. Lastly, the algorithm performs *rewiring* of the tree, i.e. it checks if any node near the most recently added one gets a lowest cost if using such node as its parent. Rewiring is the key to the asymptotic optimality of the algorithm.

In our approach, outlined in Alg. 1, the nodes V of the RRT* tree are tuples from $W_{free} \times Z$, with V rooted at (x_{init}, z_0), $x_{init} \in W_{free}$, $z_0 \in Z_0$. The function $near(x, z)$ returns every node (x', z') of the tree V such that $z' = z$ and x' is within a ball of radius $\gamma(\log(|V|)/|V|)^{1/n}$ from x, where γ is a design parameter and $|V|$ the cardinality of V.

Thanks to the controller that solves Problem 4, there always exists a control law that steers the system from a parent node x_{pa} to ϵ-neighborhood of a child node x if there exists an obstacle-free, convex polytope containing both x_{pa} and x. Therefore, let us introduce the function $poly_{traj}(x_{pa}, x)$ to construct such polytope. For simplicity, we present it here for $n = 2$, but this can be extended to any dimensions in a straightforward way.

Let us define a design parameter $\beta \gg \epsilon$, which will directly influence the size of the polytope. Let $\hat{v}$ denote the unit vector $\hat{v} = (x_{pa} - x)/||x_{pa} - x||$, and $\hat{w}$ the unit vector normal to $\hat{v}$. Then $poly_{traj}(x_{pa}, x)$ can be written as

$$\begin{bmatrix} \hat{v} \\ \hat{w} \\ -\hat{v} \\ -\hat{w} \end{bmatrix} x \leq \begin{bmatrix} \beta + ||x_{pa} - x||/2 + \hat{v}\mu \\ \beta + \hat{w}\mu \\ \beta + ||x_{pa} - x||/2 - \hat{v}\mu \\ \beta - \hat{w}\mu \end{bmatrix}, \tag{3}$$

with $\mu = (x_{pa} - x)/2$, the midpoint of the segment.

Such polytope created by $poly_{traj}$ is checked to be obstacle-free by the Boolean function *obsFree*() (line 5); if so, the *update* function (Alg. 2) is called (line 6).

The *update* function described in Alg. 2 is responsible for determining new edges to be added to the tree. Given parent x_{pa} and child x nodes, they either have different or equal set of labels according to the labeling function L.

1) $L(x_{pa}) \neq L(x)$: in such case, there must be a point x_{mid} on the trajectory from parent to child node that lies on the boundary between differently labeled workspace regions. Since, according to Definition 1, the parent's label must hold true on the entire trajectory to its child node, x_{mid} must be considered as an intermediary node between x_{pa} and x (lines 1-9). From $poly_{traj}$, two other polytopes are created: (i) its intersection with the region labeled $L(x_{pa})$ (line 2), (ii) its intersection with the region labeled $L(x)$ (line 3). Then, if they are *connected*, i.e. there is one edge common to both polytopes (line 4), x_{mid} is created at the intersection between a straight *line* connecting parent to child and the polytopes (line 5). Then, using the transition relation Δ defined by the edges E_z of the automaton, z_{mid} and z are calculated (line 6). Lastly, three edges are added to the tree (lines 7-9).

2) $L(x_{pa}) = L(x)$: described in lines 11-14, in such case the polytope is created from the intersection of $poly_{traj}$ with the region labeled $L(x_{pa})$. If the polytope is *valid*, i.e. convex (line 12), two edges are added to the tree (line 14).

Alg. 3 attempts to add an edge to the tree G connecting (x_{pa}, z_{pa}) to (x, z). In the case (x, z) is not part of the set of nodes V yet, it is added to it, an edge is created in E connecting it to (x_{pa}, z_{pa}) and the corresponding polytope π is properly stored (lines 1-3). Otherwise, if $(x, z) \in V$, an edge is created only if it yields a lower cost to reach (x, z) having (x_{pa}, z_{pa}) as its parent in comparison to the already existent edge with $parent(x, z)$ (lines 4-8). We propose to use the Euclidean distance as cost function that approximates the cost of the trajectory of 1 going through the sequence of points of a path; $cost(x_{pa}, z_{pa})$ and $cost(x, z)$ denote the current lowest cost to get from the root of the tree to x_{pa} and x, respectively, while $C(x_{pa}, x)$ calculates the Euclidean

Algorithm 1: MITL-RRT*

```
Input: $W$ - workspace, $\phi$ - mission given as Zone automaton
       $Z(\mathcal{A})$, $N$ - number of iterations
Output: p - path, z - sequence of states of $Z(\mathcal{A})$,
        $\pi$ - sequence of polytopes
1  $V \leftarrow \{(x_{init}, z_0)\}$; $E \leftarrow \emptyset$;
2  for $i = 1, \ldots, N$ do
3      $(x, z) \leftarrow sample$;
4      for $(x', z') \in near(x, z)$ do
5          if $obsFree(poly_{traj}(x', x))$ then
6              $update((x', z'), x)$
7      if $(x, z) \in V$ then
8          for $(x', z') \in near(x, z)$ do
9              if $obsFree(poly_{traj}(x, x'))$ then
10                 $update((x, z), x')$
11 if $existsSolution()$ then
12     return $(\boldsymbol{p}, \boldsymbol{z})$, $\boldsymbol{\pi}$
13 else
14     return Fail
```

Algorithm 2: $update((x_{pa}, z_{pa}), x)$

```
1  if $L(x) \neq L(x_{pa})$ then
2      $\pi_1 = poly_{traj}(x_{pa}, x) \cap L^{-1}(x_{pa})$
3      $\pi_2 = poly_{traj}(x_{pa}, x) \cap L^{-1}(x)$
4      if $connected(\pi_1, \pi_2)$ then
5          $x_{mid} = line(x_{pa}, x) \cap \pi_1$
6          $z_{mid} \leftarrow \Delta(z_{pa}, L(x))$;   $z \leftarrow \Delta(z_{mid}, L(x))$
7          $addEdge((x_{pa}, z_{pa}), (x_{mid}, z_{mid}), \pi_1)$
8          $addEdge((x_{mid}, z_{mid}), (x, z_{mid}), \pi_2)$
9          $addEdge((x_{mid}, z_{mid}), (x, z), \pi_2)$
10 else
11     $\pi = poly_{traj}(x_{pa}, x) \cap L^{-1}(x_{pa})$
12     if $valid(\pi)$ then
13         $addEdge((x_{pa}, z_{pa}), (x, z_{pa}), \pi)$
14         $addEdge((x_{pa}, z_{pa}), (x, \Delta(z_{pa}, L(x))), \pi)$
```

Algorithm 3: $addEdge((x_{pa}, z_{pa}), (x, z), \pi)$

```
1 if $(x, z) \notin V$ then
2     $V \leftarrow V \cup \{(x, z)\}$
3     $E \leftarrow E \cup \{((x_{pa}, z_{pa}), (x, z))\}$;    $\pi(x, z) = \pi$
4 else
5     if $cost(x_{pa}, z_{pa}) + C(x_{pa}, x) < cost(x, z)$ then
6         $E \leftarrow E \setminus \{(parent(x, z), (x, z))\}$
7         $E \leftarrow E \cup \{((x_{pa}, z_{pa}), (x, z))\}$
8         $\pi(x, z) = \pi$
```

distance between them.

Lastly, the *rewiring* procedure is performed (Alg. 1 lines 8-10), where it checks if any node (x', z') near the recently added node (x, z) can benefit from having it as parent. After the predetermined number N of iterations is reached, the $existsSolution()$ function performs a graph search, looking for the lowest-cost path rooted in (x_{init}, z_0) and ending at any (x_{end}, z_{end}) such that $z_{end} \in F_z$. If no path is found, it returns a failure message (lines 11-14).

Lemma 3 (Untimed word acceptance)**.** *A path* $\mathbf{p} = x_0 \dots x_\ell$ *returned by Alg. 1 (if found) generates a word* $L(x_0) \dots L(x_\ell)$ *accepted by the Zone automaton* $Z(\mathcal{A})$*.*

Proof. Given the tree $G = (V, E)$ built by Alg. 1, assume that the modified graph search on Alg. 1 Line 11 returns $(\mathbf{p}, \mathbf{z})$, a sequence of nodes $(x_0, z_0) \dots (x_\ell, z_\ell)$ in V such that $z_0 \xrightarrow{L(x_1)} z_1 \xrightarrow{L(x_2)} z_2 \dots \xrightarrow{L(x_\ell)} z_\ell$ is an accepting run of $Z(\mathcal{A})$. $\mathbf{p} = (x_0 x_1 \dots x_\ell)$ and $\mathbf{z} = (z_0 z_1 \dots z_\ell)$ are projections of $(\mathbf{p}, \mathbf{z})$ onto W_{free} and Z, respectively, and since $\mathbf{z}$ is an accepting run in $Z(\mathcal{A})$, it accepts the word $L(x_0)L(x_1) \dots L(x_\ell)$. □

According to Lemma 2, there must exist a sequence of time stamps $\mathbf{t} = t_0 t_1 \dots t_\ell$ that, associated with $L(x_0)L(x_1) \dots L(x_\ell)$, results in a timed word $w = (L(x_0), t_0)(L(x_1), t_1) \dots (L(x_\ell), t_\ell)$ such that the TA $\mathcal{A}$ accepts w.

V. Timed Path Computation

To solve Problem 3, it is necessary to determine a sequence of time intervals δ_i for $i \in I = \{1, \dots, |\mathbf{x}| - 1\}$ that together with the path $\mathbf{p}$ returned by Alg. 1 generates a timed path $(\mathbf{p}, \mathbf{t})$, with $t_0 = 0$, $t_i = t_{i-1} + \delta_i$ producing an accepting timed word $w(\mathbf{p}, \mathbf{t})$ of the TA.

From an initial state (q_0, ν_0) a run of a TA over a timed word $w(\mathbf{p}, \mathbf{t}) = (w_i, t_i)$, $i \in I$, consists of a sequence of time delays followed by discrete jumps written as

$$(q_{i-1}, \nu_{i-1}) \xrightarrow{\delta_i} (q_{i-1}, \nu_{i-1} + \delta_i) \xrightarrow{w_i} (q_i, \nu_i) \tag{4}$$

with $\nu_i = (\nu_{i-1} + \delta_i)[R := 0]$. Considering the sequence $\mathbf{z}$ of states of the Zone automaton returned by Alg. 1, such run can be written as: for every transition from $z_{i-1} = \langle q_{i-1}, \varphi_{i-1} \rangle$ to $z_i = \langle q_i, \varphi_i \rangle$, $i \in I$, one can write that

$$\langle q_{i-1}, \varphi_{i-1} \rangle \to \langle q_{i-1}, \varphi_{i-1} \Uparrow \wedge Inv(q_{i-1}) \rangle \to \langle q_i, \varphi_i \rangle. \tag{5}$$

We propose to solve the time stamp assignment problem via Linear Program (LP), which calculates time intervals δ_i directly proportional to the distance between consecutive nodes $d_{i-1,i}$, while subject to the time constraints from (4) and (5). We pose the LP as to minimize the maximum speed reached between two consecutive points on the path, assuming that the actual length of the trajectory between two points is approximated by the straight-line distance between them:

$$\min_{\delta_1, \dots, \delta_{|\mathbf{x}|-1}} \quad \xi \tag{6a}$$

$$\text{s.t.} \quad \forall i \in I, \ \frac{d_{i-1,i}}{\delta_i} \leq \xi \tag{6b}$$

$$\forall i \in I, \nu_{i-1} + \delta_i \in (\varphi_{i-1} \Uparrow \wedge Inv(q_{i-1})) \tag{6c}$$

$$\forall i \in I, \nu_i \in \varphi_i \tag{6d}$$

with $d_{i-1,i}$ the Euclidean distance between x_{i-1} and x_i. In such formulation, ξ puts an upper-bound on the relation between $d_{i-1,i}$ and δ_i (6b). In turn, (6c) ensures the time-passing transition of the TA happens within the bounds of the clock zone and (6d) does the same for the discrete jump.

Lemma 4 (Mission satisfaction)**.** *A solution* $(\mathbf{p}, \mathbf{z})$*,* $\boldsymbol{\pi}$ *returned by Alg. 1 together with the sequence of time stamps* $\mathbf{t}$ *obtained as a solution to* (6) *produce a timed word* $w(\mathbf{p}, \mathbf{t})$ *that satisfies the MITL mission* ϕ*.*

Proof. Let a timed path $(\mathbf{p}, \mathbf{t}) = (x_0, t_0) \dots (x_\ell, t_\ell)$ be defined by the projection of $(\mathbf{p}, \mathbf{z})$ onto W_{free} and $\mathbf{t} = t_0 t_1 \dots t_\ell$ that solve the LP (6) such that $t_0 = 0$ and $t_i = t_{i-1} + \delta_i$. Alg. 1 encloses the projection onto W_{free} of every edge $((x_i, z_i)(x_{i+1}, z_{i+1})) \in E$ of the tree $G = (V, E)$ with an obstacle-free, convex polytope π_i such that $L(x_i)$ holds true for any trajectory in (x_i, x_{i+1}) if there exists a control law able to maintain the system (1) inside π_i. Therefore, according to Def. 1, the timed path $(\mathbf{p}, \mathbf{t})$ generates a timed word $w(\mathbf{p}, \mathbf{t})$ such that $L(x_i)$ holds true during the entire time interval $[t_i, t_{i+1})$. Since the time stamps are calculated by the LP based on (4) and (5), a run of the TA $\mathcal{A}$ over the timed word $w(\mathbf{p}, \mathbf{t})$ is an accepting run, and therefore satisfies the bounded MITL formula ϕ as in Def. 3. □

VI. Hybrid Feedback Control Design

To solve Problem 4, we propose a hybrid feedback control strategy consisting of $|\mathbf{p}| - 1$ continuous-time feedback control laws $\boldsymbol{u}_i(\boldsymbol{x}, t)$ and a switching mechanism. With $i \in \{1, \dots, |\mathbf{p}| - 1\}$, the switching mechanism is

$$\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{u}_i(\boldsymbol{x}, t) \text{ for } t_{i-1} \leq t < t_i. \tag{7}$$

where each $\boldsymbol{u}_i(\boldsymbol{x}, t)$ is designed based on Time-varying Control Barrier Functions formulated to solve requirements stated in Signal Temporal Logic (STL) [6].

Definition 6 (STL [21])**.** *Let U be a collection of predicates $\mu(x)$, effective functions of the form $\mu : \mathbb{R}^n \to \mathbb{B}$. An STL formula is an MITL formula over the atomic propositions $\mu(x) \in U$.*

In order to drive the system starting in x_{i-1} at t_{i-1} to x_i at t_i time units while staying inside an obstacle-free convex polytope π_i, whose H-representation is $A_i \boldsymbol{x} \leq b_i$, the corresponding STL formula ψ_i is written as

$$\psi_i = \mathcal{F}_{[t_i,t_i]}(\|\boldsymbol{x}-x_i\| \leq \epsilon) \wedge \mathcal{G}_{[t_{i-1},t_i]}(A_i\boldsymbol{x} \leq B_i), \quad (8)$$

with $B_i = b_i + \epsilon$. We expand the polytopes by ϵ to ensure the ball of radius ϵ around each x_i is inside both π_i and π_{i+1}. If now $\bigwedge_{i=1}^{|\mathbf{p}|-1} \psi_i$ is satisfied, we guarantee that Problem 4 is solved. The design of $\boldsymbol{u}_i(\boldsymbol{x},t)$ then depends on ψ_i and is based on time-varying control barrier functions as in [6]. We propose control design below and refer the reader to the original paper for further intuition. First, define

$$\mathfrak{b}_i'(\boldsymbol{x},t) = \gamma_i(t) - \|\boldsymbol{x}-x_i\|$$

where $\gamma_i(t)$ is designed such that $\mathfrak{b}_i'(\boldsymbol{x},t_i) = \gamma_i(t_i) - \|\boldsymbol{x} - x_i\| \leq \epsilon - \|\boldsymbol{x}-x_i\|$. The intuition here is that the control law $\boldsymbol{u}_i(\boldsymbol{x},t)$ will enforce that $\mathfrak{b}_i'(\boldsymbol{x}(t),t) \geq 0, \forall t \in [t_{i-1},t_i)$ such that consequently $\|\boldsymbol{x}(t_i) - x_i\| \leq \epsilon$ holds, i.e. $\mathcal{F}_{[t_i,t_i]}(\|\boldsymbol{x} - x_i\| \leq \epsilon)$ is satisfied. In particular, we set $\gamma_i(t) = \frac{\epsilon - \gamma_{i,0}}{t_i - t_{i-1}}(t - t_{i-1}) + \gamma_{i,0}$ where $\gamma_{i,0}$ is a design parameter explained later with $\gamma_{i,0} > \epsilon$ such that $\dot{\gamma}_i(t) < 0$ for all $t \geq t_{i-1}$. Note that hence $\mathfrak{b}_i'(\boldsymbol{x},t_i) = \epsilon - \|\boldsymbol{x} - x_i\|$ so that $\mathfrak{b}_i'(\boldsymbol{x},t)$ accounts for the first part of the conjunction in (8). For the second part in (8), let N_{A_i} denote the number of rows in A_i and B_i and for each predicate function $j \in \{1,\ldots,N_{A_i}\}$, select

$$\mathfrak{b}_{i,j}''(\boldsymbol{x}) = B_i(j) - A_i(j,:)\boldsymbol{x}$$

where $B_i(j)$ denotes the j-th element of B_i, while $A_i(j,:)$ the j-th row of A_i. Finally, we compose the barrier function

$$\mathfrak{b}_i(\boldsymbol{x},t) = -\frac{1}{\eta_i} \ln\Big(\exp\big(-\eta_i \mathfrak{b}_i'(\boldsymbol{x},t)\big) + \sum_{j=1}^{N_{A_i}} \exp\big(-\eta_i \mathfrak{b}_{i,j}''(\boldsymbol{x})\big)\Big) \quad (9)$$

where $\eta_i > 0$ is another design parameter. It holds that $\mathfrak{b}_i(\boldsymbol{x},t) \leq \min\big(\mathfrak{b}_i'(\boldsymbol{x},t), \mathfrak{b}_{i,1}''(\boldsymbol{x}),\ldots,\mathfrak{b}_{i,N_{A_i}}''(\boldsymbol{x})\big)$ and it can be proven that $\lim_{\eta_i\to\infty} \mathfrak{b}_i(\boldsymbol{x},t) = \min\big(\mathfrak{b}_i'(\boldsymbol{x},t), \mathfrak{b}_{i,1}''(\boldsymbol{x}),\ldots,\mathfrak{b}_{i,N_{A_i}}''(\boldsymbol{x})\big)$. Consequently, when $\boldsymbol{u}_i(\boldsymbol{x},t)$ is such that $\mathfrak{b}_i(\boldsymbol{x}(t),t) \geq 0$ for all $t \in [t_{i-1},t_i)$, then, for all $t \in [t_{i-1},t_i)$, $\mathfrak{b}_i'(\boldsymbol{x}(t),t) \geq 0$ and $\mathfrak{b}_{i,j}''(\boldsymbol{x}(t)) \geq 0$ for all $j \in \{1,\ldots,N_{A_i}\}$, which implies that ψ_i is satisfied. Next, define $\mathfrak{C}_i^{\delta_i}(t) = \{\boldsymbol{x} \in \mathbb{R}^2 | \mathfrak{b}_i(\boldsymbol{x},t) \geq \delta_i\}$ where $\mathfrak{C}_i^0(t)$ is the set where the solution $\boldsymbol{x} : [t_{i-1},t_i) \to \mathbb{R}^2$ of (1) with initial position $\boldsymbol{x}(t_{i-1})$ needs to evolve in so that ψ_i is satisfied. It is assumed that the design parameters $\eta_i > 0$, $\delta_i > 0$, and $\gamma_{i,0} > \epsilon$ are selected such that $\mathfrak{C}_i^{\delta_i}(t_i) \neq \emptyset$ and $\boldsymbol{x}(t_{i-1}) \in \mathfrak{C}_i^{\delta_i}(t_{i-1})$. This can always be achieved by considering a sufficiently small δ_i and choosing η_i and $\gamma_{i,0}$ sufficiently large. Note in this respect that $A_i\boldsymbol{x}(t_{i-1}) < B_i$ and $A_i x_i < B_i$. We emphasize that $\delta_i > 0$ ensures that $\mathfrak{C}_i^0(t)$ is neither empty nor a singleton for all $t \in [t_{i-1},t_i)$, which becomes important in the proof of Corollary 1. A systematic procedure to select η_i and $\gamma_{i,0}$ is omitted due to space limitations and presented in [22]. We next propose $\boldsymbol{u}_i(\boldsymbol{x},t)$ that achieves $\mathfrak{b}_i(\boldsymbol{x}(t),t) \geq 0$ for all $t \in [t_{i-1},t_i)$. First, consider the convex quadratic program

$$\min_{\hat{\boldsymbol{u}}} \ \hat{\boldsymbol{u}}^T Q \hat{\boldsymbol{u}} \quad (10a)$$

$$\text{s.t.} \ \frac{\partial \mathfrak{b}_i(\boldsymbol{x},t)}{\partial \boldsymbol{x}}^T \big(f(\boldsymbol{x}) + g(\boldsymbol{x})\hat{\boldsymbol{u}}\big) + \frac{\partial \mathfrak{b}_i(\boldsymbol{x},t)}{\partial t} \geq \Big\|\frac{\partial \mathfrak{b}_i(\boldsymbol{x},t)}{\partial \boldsymbol{x}}\Big\| D - \alpha_i(\mathfrak{b}_i(\boldsymbol{x},t)) \quad (10b)$$

where α_i is a class $\mathcal{K}$-function and Q is a positive semidefinite matrix. Since $g(\boldsymbol{x})g(\boldsymbol{x})^T$ is positive definite, it holds that $\frac{\partial \mathfrak{b}_i(\boldsymbol{x},t)}{\partial \boldsymbol{x}}^T g(\boldsymbol{x}) = \mathbf{0}_m{}^T$ if and only if $\frac{\partial \mathfrak{b}_i(\boldsymbol{x},t)}{\partial \boldsymbol{x}} = \mathbf{0}_2$. Note that (10b) reduces to $\frac{\partial \mathfrak{b}_i(\boldsymbol{x},t)}{\partial t} \geq -\alpha_i(\mathfrak{b}_i(\boldsymbol{x},t))$ if $\frac{\partial \mathfrak{b}_i(\boldsymbol{x},t)}{\partial \boldsymbol{x}} = \mathbf{0}_2$ so that no $\hat{\boldsymbol{u}}$ appears in (10b). We next provide an assumption that relaxes [6, Assumption 3] and later, in Corollary 1, show that α_i can be selected such that this assumption holds.

Assumption 1. *For each $(\boldsymbol{x},t) \in \mathfrak{C}_i^0(t) \times [t_{i-1},t_i)$ with $\frac{\partial \mathfrak{b}_i(\boldsymbol{x},t)}{\partial \boldsymbol{x}} = \mathbf{0}_2$ it holds that $\frac{\partial \mathfrak{b}_i(\boldsymbol{x},t)}{\partial t} > -\alpha_i(\mathfrak{b}_i(\boldsymbol{x},t))$.*

Lemma 5. *Assume that ψ_i is of the form (8) and encoded in $\mathfrak{b}_i(\boldsymbol{x},t)$ according to (9) and that Assumption 1 holds, then $\boldsymbol{u}_i(\boldsymbol{x},t) = \hat{\boldsymbol{u}}$ where $\hat{\boldsymbol{u}}$ is given by (10) leads to $(\boldsymbol{x},0) \models \psi_i$.*

Proof. For cases where $\frac{\partial \mathfrak{b}_i(\boldsymbol{x},t)}{\partial \boldsymbol{x}} \neq \mathbf{0}_2$, it follows that $\frac{\partial \mathfrak{b}_i(\boldsymbol{x},t)}{\partial \boldsymbol{x}} g(\boldsymbol{x}) \neq \mathbf{0}_m{}^T$ and hence $\hat{\boldsymbol{u}}$ in (10b) can be selected so that (10) is feasible. For cases where $\frac{\partial \mathfrak{b}_i(\boldsymbol{x},t)}{\partial \boldsymbol{x}} = \mathbf{0}_2$, feasibility is guaranteed by Assumption 1. By virtue of [23, Thm. 1] it follows that the solution $\hat{\boldsymbol{u}}$ to (10) and hence $\boldsymbol{u}_i(\boldsymbol{x},t)$ is Lipschitz continuous. This implies the existence of a unique solution $\boldsymbol{x} : [t_{i-1},\tau_{i,\max}) \to \mathbb{R}^n$ to (1) with $\tau_{i,\max} > 0$. Note next that (10b) implies

$$\frac{\partial \mathfrak{b}_i(\boldsymbol{x},t)}{\partial \boldsymbol{x}}^T \big(f(\boldsymbol{x}) + g(\boldsymbol{x})\hat{\boldsymbol{u}} + \boldsymbol{d}\big) + \frac{\partial \mathfrak{b}_i(\boldsymbol{x},t)}{\partial t} \geq -\alpha_i(\mathfrak{b}_i(\boldsymbol{x},t))$$

for all $\boldsymbol{d} \in \mathfrak{D}$ since $-\frac{\partial \mathfrak{b}_i(\boldsymbol{x},t)}{\partial \boldsymbol{x}}^T \boldsymbol{d} \leq |\frac{\partial \mathfrak{b}_i(\boldsymbol{x},t)}{\partial \boldsymbol{x}}^T \boldsymbol{d}| \leq \|\frac{\partial \mathfrak{b}_i(\boldsymbol{x},t)}{\partial \boldsymbol{x}}\|\|\boldsymbol{d}\| \leq \|\frac{\partial \mathfrak{b}_i(\boldsymbol{x},t)}{\partial \boldsymbol{x}}\| D$. Consequently, the solution $\boldsymbol{x}(t)$ satisfies $\dot{\mathfrak{b}}_i(\boldsymbol{x}(t),t) \geq -\alpha_i(\mathfrak{b}_i(\boldsymbol{x}(t),t))$ for all $t \in [t_{i-1},\tau_{i,\max})$. The solution $\boldsymbol{x}(t)$ is defined over $[t_{i-1},t_i)$, i.e., $\tau_{i,\max} \geq t_i$, due to [24, Thm. 3.3] and since $\boldsymbol{x}(t)$ is forced to remain in a compact set, i.e., within the convex polytope π_i. According to [6, Thm. 1] it follows that $\mathfrak{b}(\boldsymbol{x}(t),t) \geq 0$ for all $t \in [t_{i-1},t_i)$ so that $(\boldsymbol{x},0) \models \psi_i$. □

We next show that Assumption 1 can be satisfied by a suitable choice of α_i.

Corollary 1. *The class $\mathcal{K}$-function α_i can be selected such that Assumption 1 holds.*

Proof. Note that the function $\mathfrak{b}_i(\boldsymbol{x},t')$ is concave in $\boldsymbol{x}$ for each fixed $t' \in [t_{i-1},t_i)$. This follows since the functions $\mathfrak{b}_i'(\boldsymbol{x},t')$ and $\mathfrak{b}_{i,j}''(\boldsymbol{x})$ are concave. Then due to [25, Section 3.5] it holds that $\exp(-\eta_i\mathfrak{b}_i'(\boldsymbol{x},t'))$ and $\exp(-\eta_i\mathfrak{b}_{i,j}''(\boldsymbol{x}))$ are log-convex. It further holds that a sum of log-convex functions is log-convex. Consequently, it holds that $-\frac{1}{\eta_i}\ln\big(\exp(-\eta_i\mathfrak{b}_i'(\boldsymbol{x},t')) + \sum_{j=1}^{N_{A_i}} \exp(-\eta_i\mathfrak{b}_{i,j}''(\boldsymbol{x}))\big)$ is concave. This implies that, for each $t' \in [t_{i-1},t_i)$, there is only one optimum of $\mathfrak{b}(\boldsymbol{x},t')$, i.e., there is $\boldsymbol{x}_{t'}^* = \text{argmax}_{\boldsymbol{x}\in\mathbb{R}^n} \mathfrak{b}(\boldsymbol{x},t')$ with $\boldsymbol{x}_{t'}^* \in \mathfrak{C}_i^0(t')$ and $\mathfrak{b}(\boldsymbol{x}_{t'}^*,t') > \mathfrak{b}(\boldsymbol{x},t')$ for all $\boldsymbol{x} \neq \boldsymbol{x}_{t'}^*$ such that $\frac{\partial \mathfrak{b}_i(\boldsymbol{x},t')}{\partial \boldsymbol{x}} = \mathbf{0}_2$ if and only if $\boldsymbol{x} = \boldsymbol{x}_{t'}^*$. Note next that $\mathfrak{b}_i(\boldsymbol{x},t)$ is constructed such that $\mathfrak{b}_i(\boldsymbol{x}_{t'}^*,t') \geq \delta_i > 0$ for each $t' \in [t_{i-1},t_i)$ and hence it holds that $\mathfrak{b}_i'(\boldsymbol{x}_{t'}^*,t') \geq \delta_i$. Furthermore, note that $\mathfrak{b}_i'(\boldsymbol{x},t')$ and each $\mathfrak{b}_{i,j}''(\boldsymbol{x})$ are continuously differentiable so that these functions are upper bounded within the convex polytope π_i, i.e., there exists a $\mathfrak{b}_i^{\max} > 0$ so that $\max(\mathfrak{b}_i'(\boldsymbol{x},t'), \mathfrak{b}_{i,1}''(\boldsymbol{x}),\ldots,\mathfrak{b}_{i,N_{A_i}}''(\boldsymbol{x})) \leq \mathfrak{b}_i^{\max}$. Hence we derive that, for each $t' \geq t_{i-1}$, it holds that

$$\frac{\partial \mathfrak{b}(\boldsymbol{x}_{t'}^*,t')}{\partial t} = \frac{-\exp(-\eta_i\mathfrak{b}_i'(\boldsymbol{x}_{t'}^*,t'))|\frac{\partial \mathfrak{b}_i'(\boldsymbol{x}_{t'}^*,t')}{\partial t}|}{\exp(-\eta_i\mathfrak{b}_i'(\boldsymbol{x}_{t'}^*,t')) + \sum_{j=1}^{N_{A_i}} \exp(-\eta_i\mathfrak{b}_{i,j}''(\boldsymbol{x}_{t'}^*,t'))}$$

$$\geq \frac{-\exp(-\eta_i \delta_i)|\frac{\epsilon-\gamma_{i,0}}{t_i-t_{i-1}}|}{(N_{A_i}+1)\exp(-\eta_i \mathfrak{b}_i^{\max})} = \zeta_i.$$

where ζ_i is in particular independent of t' and $\boldsymbol{x}^*_{t'}$. If it is now guaranteed that $\zeta_i > -\alpha_i(\delta_i)$, then it holds that $\frac{\partial \mathfrak{b}(\boldsymbol{x}^*_{t'}, t')}{\partial t} > -\alpha_i(\mathfrak{b}_i(\boldsymbol{x}^*_{t'}, t')$ for all $t' \in [t_{i-1}, t_i)$, which thus ensures that Assumption 1 holds. In fact, by the choice of $\alpha_i(\delta_i) = \kappa_i \delta_i$, we can select $\kappa_i > \frac{-\zeta_i}{\delta_i}$ such that this is the case. □

VII. Correctness

In this section, we analyze the properties of collision avoidance and probabilistic completeness, and discuss how our approach addresses Problem 1.

Theorem 1 (Collision avoidance)**.** *The proposed approach ensures collision avoidance with obstacles.*

Proof. Assume a system (1) under a time-bounded MITL mission ϕ (2) is controlled using the approach proposed in Sec. III. The motion planner presented in Sec. IV creates a tree $G = (V, E)$ whose edges' projections on W_{free} are enclosed by obstacle-free, convex polytopes. The control law (7) designed in Sec. VI ensures the trajectory of the system on the workspace W to be restricted to a subset formed by only the union of these polytopes $\bigcup \boldsymbol{\pi} \subset W_{free} \subseteq W$. □

Let $\varepsilon(x)$ define a ball of radius ϵ centered in x. Assume the timed path $(\mathbf{p}, \mathbf{t}) = (x_0, t_0)(x_1, t_1) \dots (x_\ell, t_\ell)$ and the sequence of obstacle-free, convex polytopes $\boldsymbol{\pi}$ solve Problems 2 and 3 for a mission specification ϕ in bounded MITL. Also assume there exists a control law $\boldsymbol{u}(\boldsymbol{x}, t)$ that solves Problem 4. Such control law takes the system from its initial configuration $(x_0, t_0 = 0)$ and, while maintaining it inside the obstacle-free workspace defined by π_1, drives it to (χ_1, t_1) with $\chi_1 \in \varepsilon(x_1)$. Therefore, the actual resulting timed path of the system is $(x_0, t_0)(\chi_1, t_1)(\chi_2, t_2)(\chi_3, t_3) \dots (\chi_\ell, t_\ell)$ for $\chi_i \in \varepsilon(x_i)$. In fact, ϵ can be chosen to be arbitrarily small so that the actual timed path is almost equal to $(\mathbf{p}, \mathbf{t})$.

Theorem 2 (Probabilistic completeness)**.** *The motion planning approach proposed in Sec. IV is probabilistically complete.*

Proof. The proof follows the probabilistic completeness proof of the RRT* algorithm [8] with a minor modification. Let $\delta > \beta\sqrt{2}$ be a real number, where β is directly related to the size of the polytopes as in (3). If the problem is robustly feasible, i.e. there exists a path with strong δ-clearance, a solution is found with probability one as the number of vertices in the tree reaches infinity. Note that $\delta > 0$ in [8]; modifying it to $\delta > \beta\sqrt{2}$ ensures that if there exists a path with strong δ-clearance, a sequence of polytopes as in (3) will fit in it, i.e. $\bigcup \boldsymbol{\pi} \subset W_{free} \subseteq W$. □

VIII. Simulations

We demonstrate our approach in an illustrative case study, implemented in Python 2.7, and performed on an Intel®Core$^{\text{TM}}$ i7-7700HQ CPU with 2.8GHz clock speed and 16GB RAM under Ubuntu 16.04.

Consider a dynamical system with coupled-input given by

$$\dot{x}_1 = u_1 - 0.5u_2, \quad \dot{x}_2 = u_2$$

deployed in an office-like environment, with obstacles that resembles tables and walls and two goal regions A and B, that is subject to two different mission specifications

$$\phi_1 = \mathcal{F}_{[5,10]}A \wedge \mathcal{F}_{[15,20]}B; \qquad \phi_2 = \mathcal{G}_{[20,30]}A \wedge \mathcal{G}_{[45,60]}B.$$

Figure 2a shows the resulting trajectory of the system on the workspace when subject to ϕ_1, with the corresponding waypoints and polytopes. Due to the fact that the control law minimizes the control effort (10), two waypoints are not connected by straight lines in the resulting trajectory of the system. Figure 2b presents the evolution of such trajectory in relation to time. The same is presented for mission ϕ_2 in Figures 2c and 2d.

We ran MITL-RRT* for 3000 iterations, with $\beta = 0.2$ and $\epsilon = 0.001$, in both ϕ_1 and ϕ_2; the former took 481s to compute, generated a tree with 1019 nodes, from which 30 nodes form the satisfying path; the latter took 430s to compute, generated a tree with 952 nodes, and a path with 32 nodes. The LP (6) took 23ms to calculate time stamps for ϕ_1, and 122ms for ϕ_2. On average, a control loop (10) took 0.711ms.

Although the specifications ϕ_1 and ϕ_2 might look similar as they both require the system to first reach A, then B (Figs. 2a and 2c), comparing them highlights how our approach is capable of generating correct timed paths (Figs. 2b and 2d).

IX. Conclusion

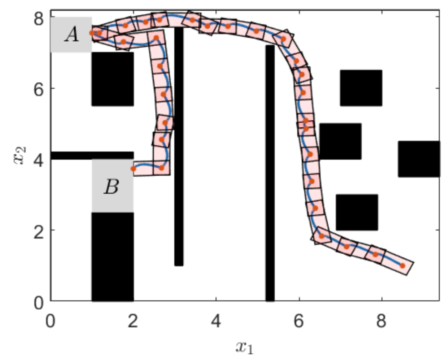
This paper presented an approach to integrate motion planning and control in order to enforce a system to satisfy a mission specification defined using Metric Interval Temporal Logic (MITL). Such specification is translated into Timed Automaton and then to its Zone automaton, which is used in the sampling-based algorithm MITL-RRT* to find a path on the obstacle-free workspace that is accepted by the Zone automaton. A sequence of time-stamps, calculated via Linear Program, and of obstacle-free, convex polytopes are associated with the optimal path and fed into a hybrid feedback control law based on Time-varying Control Barrier Functions, which in turn guarantees the system "almost" satisfies the mission.

Future work comprises of (i) replacing MITL formalism with Signal Temporal Logic (STL) so that more complex missions can be easily described, e.g. "Visit region A within 10 to 20 time units while always maintaining at least 50cm away from any obstacle"; and (ii) supporting multi-agent systems.

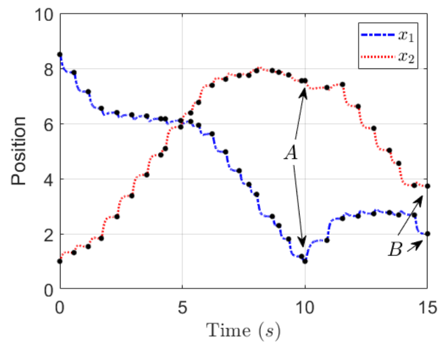
References

[1] J. A. DeCastro and H. Kress-Gazit, "Synthesis of nonlinear continuous controllers for verifiably correct high-level, reactive behaviors," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 378–394, 2015.

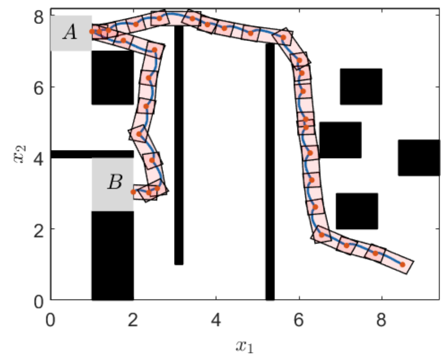
[2] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.



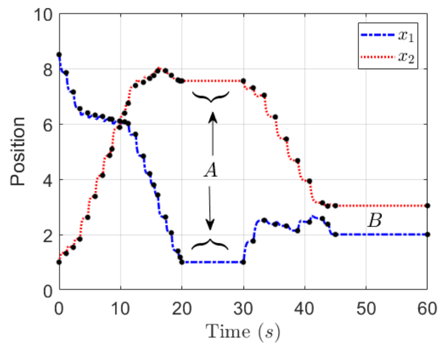
(a)



(b)



(c)



(d)

Fig. 2. (a) and (c) present the resulting trajectories, in blue, for $\phi_1 = \mathcal{F}_{[5,10]}A \wedge \mathcal{F}_{[15,20]}B$ and $\phi_2 = \mathcal{G}_{[20,30]}A \wedge \mathcal{G}_{[45,60]}B$, respectively. The system starts from the bottom right of each figure, and the controller drives it throughout the waypoints marked in red dots without leaving the polytopes. (b) and (d) show the system evolution in time when subject to ϕ_1 and ϕ_2, respectively. The black dots represent the projection of the timed path onto x_1 and x_2.

[3] S. Andersson, A. Nikou, and D. V. Dimarogonas, "Control synthesis for multi-agent systems under metric interval temporal logic specifications," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 2397–2402, 2017.

[4] Y. Zhou, D. Maity, and J. S. Baras, "Timed automata approach for motion planning using metric interval temporal logic," in *Control Conference (ECC), 2016 European*. IEEE, 2016, pp. 690–695.

[5] C. Vasile, V. Raman, and S. Karaman, "Sampling-based synthesis of maximally-satisfying controllers for temporal logic specifications," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 3840–3847.

[6] L. Lindemann and D. V. Dimarogonas, "Control barrier functions for signal temporal logic tasks," *IEEE Control Systems Letters*, vol. 3, no. 1, pp. 96–101, 2019.

[7] S. Karaman and E. Frazzoli, "Sampling-based motion planning with deterministic μ-calculus specifications," in *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*. IEEE, 2009, pp. 2222–2229.

[8] ——, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[9] L. I. R. Castro, P. Chaudhari, J. Tumova, S. Karaman, E. Frazzoli, and D. Rus, "Incremental sampling-based algorithm for minimum-violation motion planning," in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*. IEEE, 2013, pp. 3217–3224.

[10] C. Vasile, J. Tumova, S. Karaman, C. Belta, and D. Rus, "Minimum-violation scLTL motion planning for mobility-on-demand," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1481–1488.

[11] F. J. Montana, J. Liu, and T. J. Dodd, "Sampling-based reactive motion planning with temporal logic constraints and imperfect state information," in *Critical Systems: Formal Methods and Automated Verification*. Springer, 2017, pp. 134–149.

[12] B. Hoxha and G. Fainekos, "Planning in dynamic environments through temporal logic monitoring," in *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[13] P. Wieland and F. Allgöwer, "Constructive safety using control barrier functions," in *Proceedings of the 7th IFAC Symposium on Nonlinear Control Systems*, Pretoria, South Africa, August 2007, pp. 462–467.

[14] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017.

[15] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[16] P. Glotfelter, J. Cortés, and M. Egerstedt, "Nonsmooth barrier functions with applications to multi-robot systems," *IEEE control systems letters*, vol. 1, no. 2, pp. 310–315, 2017.

[17] R. Alur, T. Feder, and T. A. Henzinger, "The benefits of relaxing punctuality," *Journal of the ACM (JACM)*, vol. 43, no. 1, pp. 116–146, 1996.

[18] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical computer science*, vol. 126, no. 2, pp. 183–235, 1994.

[19] J. Bengtsson and W. Yi, *Timed Automata: Semantics, Algorithms and Tools*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 87–124.

[20] T. Brihaye, G. Geeraerts, H.-M. Ho, and B. Monmege, "MightyL: A compositional translation from MITL to timed automata," in *International Conference on Computer Aided Verification*. Springer, 2017, pp. 421–440.

[21] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Proceedings of the International Conference on FORMATS-FTRTFT*, Grenoble, France, September 2004, pp. 152–166.

[22] L. Lindemann and D. V. Dimarogonas, "Decentralized control barrier functions for coupled multi-agent systems under signal temporal logic tasks," in *European Control Conference*, Naples, Italy, June 2019.

[23] B. Morris, M. J. Powell, and A. D. Ames, "Sufficient conditions for the lipschitz continuity of qp-based multi-objective control of humanoid robots," in *Proceedings of the Conference on Decision and Control (CDC)*, Florence, Italy, December 2013, pp. 2920–2926.

[24] H. K. Khalil, *Nonlinear Systems*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[25] S. Boyd and L. Vandenberghe, *Convex optimization*, 1st ed. New York, NY: Cambridge University press, 2004.