Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Optimization-based Motion Planning and Runtime Monitoring for Robotic Agent with Space and Time Tolerances [⋆]

Zhenyu Lin [∗] **John S. Baras** [∗]

[∗] *Department of Electrical and Computer Engineering and the Institute for Systems Research, University of Maryland, College Park, MD 20742, USA (e-mail: zlin88@umd.edu, baras@umd.edu).*

Abstract: We present an optimization-based approach for robot planning, monitoring and self-correction problems under signal temporal logic specifications (STL). The STL specifications are translated into mixed-integer linear constraints, and we generate the reference trajectory by solving a mixed-integer-linear-programming (MILP) to maximize the overall space and time tolerances. During runtime execution, a prediction module is constantly evaluating the robustness degree of the predicted trajectory, and a self-correction module based on event-triggered model predictive control (MPC) has been designed to predict and correct possible future violations of the specifications. Simulation results show that with our approach, the robotic agent is able to generate a path that satisfies the STL specifications while maximizing space and time tolerances, and able to make corrections when there are possible violations of the specifications during runtime execution.

Keywords: Motion Planning, Signal Temporal Logic, Space and Time Tolerances, Optimization

1. INTRODUCTION

Motion and task planning for autonomous robotic agents is important in many real, physical world applications. Robotic agents have been deployed for agriculture research, surveillance, and search and rescue operations. In recent years, a new approach to the task planning problem for robotic agents has evolved by formulating system specifications in temporal logics Smith et al. (2010) Bhatia et al. (2011) Wolff et al. (2014). Linear temporal logic (LTL) allows one to specify more complicated mission tasks that are hard to express and to achieve by conventional methods Lamport (1994).

However, LTL specifications do not emphasize finite time constraints. For real applications, a robot might be required to perform a specific task within a certain time bound, rather than at some arbitrary time in the future. Metric temporal logic (MTL) Alur et al. (1996) and Signal Temporal Logic (STL) Donze and Maler (2010) have been introduced for motion planning with bounded time constraints. Task planning with bounded time constraints has been investigated in Zhou et al. (2015) and Vasumathi et al. (2014) by solving a MILP problem. An automata based approach for task planning with MTL specifications has been considered in Lin and Baras (2019).

STL allows the specification of properties of dense-time and real-valued signals. One main advantage of STL is the quantitative semantics which, in addition to the yes/no answer to the satisfaction question, provide a real number that grades the quality of the satisfaction or violation. This robustness information could be useful in motion planning, since we normally want the robotic agents to stay far away from the obstacles, and also to stay close to the center of the locations of interest. Motion planning problems with STL have been investigated in Plaku and Karaman (2016) Lindemann et al. (2018), and authors in Vasumathi et al. (2014) and Lindemann and Dimarogonas (2017) have used STL for control synthesis together with Model Predictive Control (MPC).

For autonomous systems operating in dynamic environments, the safety of motion and time requirements for the task are critical. Due to the uncertainty in the environment, the planning results obtained with respect to the system and environment models at design-time might not be transferable to the system behavior at run time. Therefore, allowing both space and time tolerances in the planning phase, and the ability of runtime monitoring and self-correction are essential. Monitoring problems for STL have been discussed in Akazaki and Ichiro (2015), Fainekos and Pappas (2009) and Donze and Maler (2010). Vasumathi et al. (2014) discusses MPC for signal temporal logic specifications, but their approach requires solving the MILP problem at each time step, and is not able to address the time robustness issue. Lindemann and Dimarogonas (2017) considers the motion planning problem using STL and introduces the Discrete Average Space Robustness to maximize the space robustness. However, in their work time robustness is not considered in the planning phase. To address these issues, in this work we divide the problem into the following two parts: (1) offline control synthesis, (2) online monitoring and self-correction. We first generate a path that considers both space and time robustness, and

[⋆] This work was partially supported by the ONR grant N00014-17-1-2622.

then we design a prediction module and event-triggered MPC module for the monitoring and self-correction.

The contributions of this paper are as follows. First, we transform the robot planning problem under signal temporal logic specifications into a mixed-integer linear programming problem, while considering both the time tolerances and space tolerances. To the best of our knowledge, this is the first work that considers both space and time tolerances in the planning phase. Second, we have designed a monitoring and self-correction framework for the runtime execution. A predicted trajectory is generated at each time step, and we propose an event-triggered model predictive control framework such that the robot is able to make self-corrections when there is predicted error during runtime execution.

The rest of the paper is organized as follows. Section 2 provides some definitions and preliminaries related to STL. Section 3 discusses the problem formulation and our method for maximum space-time tolerance planning. Section 4 presents the method of converting STL specifications into MILP. Runtime monitoring and self-correction is discussed in section 5. Section 6 shows the case studies and we conclude in section 7.

2. PRELIMINARIES

Definition 2.1. An atomic proposition is a statement about the system variables (x) that is either True($\top$) or False($\bot$) for some given values of the state variables Karaman et al. (2008).
Definition 2.2. (STL semantics) The syntax of Metric Temporal Logic (MTL) formulas are defined according to the following grammar rules:

$$\varphi ::= \mathrm{T} \,|\, \pi \,|\, \neg\varphi_1 \,|\, \varphi_1 \wedge \varphi_2 \,|\, \Box_I \varphi_1 \,|\, \varphi_1 \,\mathcal{U}_I\, \varphi_2 \,| \qquad (1)$$

where I $\subseteq$ [0, ∞]. $\mathcal{U}_I$ symbolizes the timed Until operator. Sometimes we will represent $\mathcal{U}_{[0,\infty]}$ by $\mathcal{U}$. Other Boolean and temporal operators such as conjunction ($\vee$), eventually within I ($\Diamond_I$) etc. can be represented using the grammar described in the definition.

For any signal s, let s_t denote the value of s at time t and let $(s,t) = s_t s_{t+1} s_{t+2} \cdots$ be the part of the signal that is a sequence of $s_t{}'$ for $t' \in [t, \infty)$. Accordingly, the Boolean semantics of STL is recursively defined as follows:

- $(s,t) \models (f(s) < d) \Leftrightarrow f(s_t) < d$,
- $(s,t) \models \neg(f(s) < d) \Leftrightarrow \neg((s,t) \models (f(s) < d))$,
- $(s,t) \models \varphi_1 \wedge \varphi_2 \Leftrightarrow (s,t) \models \varphi_1 \quad \text{and} \quad (s,t) \models \varphi_2$,
- $(s,t) \models \varphi_1 \vee \varphi_2 \Leftrightarrow (s,t) \models \varphi_1 \quad \text{or} \quad (s,t) \models \varphi_2$,
- $(s,t) \models \Box_{[a,b]}\varphi \Leftrightarrow (s,t') \models \varphi \quad \forall t' \in [t+a, t+b]$,
- $(s,t) \models \Diamond_{[a,b]}\varphi \Leftrightarrow \exists t' \in [t+a, t+b] \quad \text{s.t} \quad (s,t') \models \varphi$.

Definition 2.3. (Space Robustness) STL is endowed with a metric called robustness degree Donze and Maler (2010) (also called "degree of satisfaction") that quantifies how well a given signal s satisfies a given formula φ. The robustness degree is calculated recursively according to the quantitative semantic:

- $r(s, (f(s) < d), t) = d - f(s_t)$,
- $r(s, \neg(f(s) < d), t) = -r(s, (f(s) < d), t)$,
- $r(s, \varphi_1 \wedge \varphi_2, t) = \min(r(s, \varphi_1, t), r(s, \varphi_2, t))$,
- $r(s, \varphi_1 \vee \varphi_2, t) = \max(r(s, \varphi_1, t), r(s, \varphi_2, t))$,
- $r(s, \Diamond_{[a,b]}\varphi, t) = \max\limits_{t' \in [t+a, t+b]} r(s, \varphi, t')$,
- $r(s, \Box_{[a,b]}\varphi, t) = \min\limits_{t' \in [t+a, t+b]} r(s, \varphi, t')$,

Definition 2.4. (Time Robustness) The left and right time robustness of an STL formula φ with respect to a trace s at time t are defined as follows

- $\theta^-(s, f(s), t) = \max(d \geq 0 \text{ s.t.} \forall t' \in [t-d, t], (s,t) \models \varphi \Leftrightarrow (s,t') \models \varphi)$
- $\theta^+(s, f(s), t) = \max(d \geq 0 \text{ s.t.} \forall t' \in [t, t+d], (s,t) \models \varphi \Leftrightarrow (s,t') \models \varphi)$

The time robustness indicates how much the signal could be shifted to the left (right) such that the specification is still satisfied.
Assumption 2.1. (Double Integrator dynamics) The dynamics of the robot is assumed to be given by the following model:

$$X(k+1) = A \cdot X(k) + B \cdot U(k) \qquad (2)$$

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} (\Delta t^2/2) & 0 \\ 0 & (\Delta t^2/2) \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \qquad (3)$$

$$\mathbf{X}(t) = \begin{bmatrix} x(t) & y(t) & \dot{x}(t) & \dot{y}(t) \end{bmatrix}^T \qquad (4)$$

where $x(t), y(t)$ are the cartesian position of the robotic agent, and $\dot{x}(t), \dot{y}(t)$ are the velocities on each direction respectively. Let us denote the trajectory of the system starting at t_0 with initial condition x_0 and input $u(t)$ as $\mathbf{X}_{t0}^{x0,u} = \{\mathbf{X}(s) | s \geq t_0, \mathbf{X}(t+1) = f(t, \mathbf{X}(t), u(t)), \mathbf{X}(t_0) = x_0\}$. For brevity, we will use $\mathbf{X}_{t_0}$ instead of $\mathbf{X}_{t0}^{x0,u}$ whenever we do not need the explicit information about $u(t)$ and x_0. Satisfaction of a temporal specification φ by a trajectory $\mathbf{X}_{t_0}$ will be denoted as $\mathbf{X}_{t_0} \models \varphi$.

3. MAXIMUM SPACE-TIME TOLERANCES PLANNING

Space and time tolerances are important for the planning problem. With large space tolerances, the robot has a higher chance to satisfy the temporal logic specifications when the trajectory deviates from the planning path. With large time tolerances, the robot could handle the situation when the execution is slower or faster than the plan. Therefore, it is important to take both space and time tolerances into consideration. As shown in Fig 1, if space and time tolerances are not taken into considerations, all three signals are considered as satisfying $\Diamond_{[a,b]}(x > 0)$ from $t = 0$ at the same degree. However, it is clear that the space tolerance of ω_2 is small (the specification will be violated if we disturb x a little) and the time tolerance for ω_3 is small (the specification will be violated if we shift the signal a little to the right).

The planning problem considered in this paper is to determine the optimal trajectories such that the given temporal logic specifications are satisfied, and maximizing

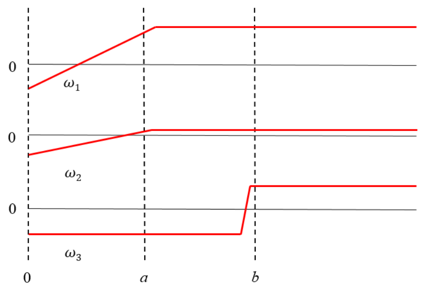

Fig. 1. Limitations of the point-wise quantitative semantics: signals ω_1, ω_2 and ω_3 are considered as satisfying $\Diamond_{[a,b]}(x > 0)$ from $t = 0$ at the same degree.

the space and time tolerances at the same time. The optimization problem could be formulated as follows:
Problem 1.

$$\begin{aligned} \max_{\boldsymbol{X}(t),u(t)} \quad & \lambda_1 r^{\varphi}_{time}(\boldsymbol{X}_{t_0}) + \lambda_2 r^{\varphi}_{space}(\boldsymbol{X}_{t_0}) \\ \textit{subject to} \quad & \boldsymbol{X}(t+1) = f(\boldsymbol{X}(t), u(t)) \\ & u_{min} \le u(t) \le u_{max} \\ & \boldsymbol{X}_{t_0} \models \varphi \end{aligned} \tag{5}$$

- $r^{\varphi}_{time}(\mathbf{X}_{t_0})$ and $r^{\varphi}_{space}(\mathbf{X}_{t_0})$ are the time and space tolerances for the trajectory $\mathbf{X}_{t_0}$ over specification φ.
- The system dynamics are given in equation (2), and the control inputs are bounded to $[u_{min}, u_{max}]$.
- $\mathbf{X}_{t_0} \models \varphi$ is the constraint that the STL specifications are satisfied.

The temporal logic constraints φ is transformed into linear constraints and will be described in details in the next section. The objective function we consider here is to maximize the overall space and time tolerances, which is

$$r^{\varphi}(\mathbf{X}_{t_0}) = \lambda_1 \cdot r^{\varphi}_{time}(\mathbf{X}_{t_0}) + \lambda_2 \cdot r^{\varphi}_{space}(\mathbf{X}_{t_0}) \tag{6}$$

where λ_1 and λ_2 are the weight coefficients, and $\lambda_1 + \lambda_2 = 1$. $r_{space}(\mathbf{X}_{t_0})$ is the space tolerance, which is defined similarly as the space robustness in Definition 2.3. $r_{time}(\mathbf{X}_{t_0})$ is the time tolerance, and our goal is to generate a trajectory that is robust to time shifting. Therefore, we propose to extend the definition of time robustness as follows:

For eventually $(\Diamond_{a,b}A)$ operator, the time tolerance is defined as follows:

$$r^{\varphi}_{time}(\mathbf{X}_{t_0}) = \sum_{t=t_a}^{t_b} \cdot \frac{1}{\sigma\sqrt{2\pi}} e^{(t-\frac{t_a+t_b}{2})^2/2\sigma^2} \cdot P^A_t \tag{7}$$

where σ is a user-defined parameter indicating the standard deviation. Basically, we want to maximize the time that the robotic agent is staying within the locations of interests, and preferably in the middle of the allowed time interval. P^A_t is a binary variable and the value is 1 when the robot is within location A at time t and it is 0 otherwise. More detail discussion on how to construct P^A_t will be described in the next section.

For always $(\Box_{[a,b]}A)$ operator, the time tolerance is defined as follows:

$$\begin{aligned} r^{\varphi}_{time}(\mathbf{X}_{t_0}) = & \sum_{t=t_a-\tau}^{t_b} \cdot \frac{1}{\sigma\sqrt{2\pi}} e^{(t-t_a)^2/2\sigma^2} \cdot P^A_t \\ & + \sum_{t=t_b}^{t_b+\tau} \cdot \frac{1}{\sigma\sqrt{2\pi}} e^{(t-t_b)^2/2\sigma^2} \cdot P^A_t \end{aligned} \tag{8}$$

where τ is a user-defined parameter and we want the robot to also satisfy the specification before t_a and after t_b. Note that the satisfaction of the specifications is already enforced by the constraint $\mathbf{X}_{t_0} \models \varphi$, and we are maximizing the space and time tolerances in the objective function based on that.

4. MIXED INTEGER LINEAR PROGRAMMING

In this section, we demonstrate our approach to translate a time-bounded temporal logic formula (constraint $\mathbf{X}_{t_0} \models \varphi$ in equation (1)) to mixed integer linear constraints on state variables and inputs. Any convex polygon can be represented as an intersection of several halfspaces. If the area of interest has a non-convex shape, we could always decompose the polygon to convex ones and link them using disjunction operators. A halfspace is expressed by a set of points, $\mathcal{H} = \{x : h_i^T x \le k_i\}$. Thus, $x(t) \in \mathcal{P}$ is equivalent to $x(t) \in \cap_{i=1}^{n} \mathcal{H}(i)$. In order to translate the temporal constraints with location atomic propositions into mixed integer convex (linear) constraints, we use a similar method as discussed in Zhou et al. (2015).

In a polygonal environment, atomic propositions (AP), $p \in \Pi$, can be related to states of the system using disjunction and conjunction of halfspaces. In other words, the relationship between measured outputs such as the location of the robotic agent and the halfspaces defines the propositions used in the temporal logic. Consider the convex polygon case and let $z_i^t \in \{0, 1\}$ be the binary variables associated with halfspaces $\{x(t) : h_i^T x \le k_i\}$ at time $t = 0, \cdots, N$. We enforce the following constraint $z_i^t = 1$ if and only if $h_i^T x \le k_i$ by adding the convex (linear) constraints,

$$h_i^T x \le k_i + M(1 - z_i^t), \quad h_i^T x \ge k_i - M z_i^t + \epsilon \tag{9}$$

where M is a large positive number and ϵ is a small positive number. If we denote $P_t^{\mathcal{P}} = \wedge_{i=1}^{n} z_i^t$, then $P_t^{\mathcal{P}} = 1$ if and only if $x(t) \in \mathcal{P}$ at time t, and 0 otherwise. Therefore, $P_t^{\mathcal{P}}$ is the binary variable that indicates whether the robotic agent lie in area $\mathcal{P}$ at time t. Let p and q denote labels for some location in the environment. The following Boolean operators, such as $\neg$, $\wedge$, $\vee$, can be translated into linear constraints. For $t \in \{0, 1, ..., N\}$, we denote the variables associated with formula φ made up with propositions $p \in \Pi$ at time t as P_t^{φ}. The next subsection will discuss the construction of P_t^{φ} for different temporal logic specifications.

4.1 MTL to Mixed Integer Linear Constraints

Let p and q denote labels for some locations in the environment.

- The negation operation, $\varphi = \neg p$ is modeled as

$$P_t^{\varphi} = 1 - P_t^p \tag{10}$$

- The conjunction operation, $\varphi = \wedge_{i=1}^{m} p_i$ is modeled as
$$P_t^{\varphi} \leq P_t^{p_i}, i = 1, \cdots m, \quad P_t^{\varphi} \geq 1 - m + \sum_{i=1}^{m} P_t^{p_i} \tag{11}$$
- The disjunction operation, $\varphi = \vee_{i=1}^{m} p_i$ is modeled as
$$P_t^{\varphi} \geq P_t^{p_i}, i = 1, \cdots m; \quad P_t^{\varphi} \leq \sum_{i=1}^{m} P_t^{p_i} \tag{12}$$

Similarly, the temporal operators can be modeled using linear constraints as well. Let $t \in \{0, 1, \cdots, N - t_2\}$, where $[t_1, t_2]$ is the time interval used in the MTL.

- Eventually: $\varphi = \Diamond_{[t_1,t_2]} p$ is equivalent to
$$\begin{aligned} &P_t^{\varphi} \geq P_{\tau}^{p}, \tau \in \{t + t_1, \cdots, t + t_2\} \\ &P_t^{\varphi} \leq \sum_{\tau=t+t_1}^{t+t_2} P_{\tau}^{p} \end{aligned} \tag{13}$$
- Always: $\varphi = \Box_{[t_1,t_2]} p$ is equivalent to
$$\begin{aligned} &P_t^{\varphi} \leq P_{\tau}^{p}, \tau \in \{t + t_1, \cdots, t + t_2\} \\ &P_t^{\varphi} \geq \sum_{\tau=t+t_1}^{t+t_2} P_{\tau}^{p} - (t_2 - t_1) \end{aligned} \tag{14}$$
- Until: $\varphi = p\,\mathcal{U}_{[t_1,t_2]}\, q$ is equivalent to
$$\begin{aligned} &a_{tj} \leq P_j^{q}, j \in \{t + t_1, \cdots, t + t_2\} \\ &a_{tj} \leq P_k^{p}, k \in \{t, \cdots, j - 1\}, j \in \{t + t_1, \cdots, t + t_2\} \\ &a_{tj} \geq P_j^{q} + \sum_{k=t}^{j-1} P_k^{p} - (j - t), j \in \{t + t_1, \cdots, t + t_2\} \\ &P_t^{\varphi} \leq \sum_{j=t+t_1}^{t+t_2} a_{tj}, \quad P_t^{\varphi} \geq a_{tj}, j \in t + t_1, \cdots, t + t_2 \end{aligned} \tag{15}$$

For the until operator, we define extra slack variables similar to Karaman et al. (2008) in order to make the constraints linear in terms of the variables. The constraints for the until operator could be interpreted as follows:

$$P_t^{\varphi} = \bigvee_{j=t+t_1}^{t+t_2} (\wedge_{k=t}^{k=j-1} P_k^{p}) \wedge P_j^{q})$$

Using this approach, we translate the given high level specification in STL ($\mathbf{X}_{t_0} \models \varphi$) to a set of mixed integer linear constraints. At the end, we add the constraint $P_0^{\varphi} = 1$, i.e. the overall specification φ is satisfied. Since Boolean variables are only introduced when halfspaces are defined, the computation cost of MILP is at most exponential to the number of halfspaces times the discrete steps N.

5. RUNTIME MONITORING AND SELF-CORRECTION

Let N be the horizon of the planning trajectory, and let $X_r(t)$ and $U_r(t)$ be the reference states and control inputs for $t \in [1, N]$ respectively. Note that $X_r(t)$ and $U_r(t)$ could be obtained offline by solving the MILP in Problem 1. During runtime, two threshold parameters θ_{space} and θ_{time} are defined to monitor the runtime execution. θ_{space} and θ_{time} are the space and time tolerances we want to maintain for the execution sequence. At time t', we denote the observed states as $X_o(t)$, where $t \in [1, t']$. The predicted states $X_p(t)$ of the robot is generated based on the observed states and the reference inputs until the end of the execution ($t = N$), i.e,

$$\begin{aligned} X_p(\tau + 1) = f(X_p(\tau), U_r(\tau)), \quad \tau = t', \cdots N - 1 \\ X_p(\tau) = X_o(\tau) \quad \text{for} \quad \tau = 1, \cdots, t' \end{aligned} \tag{16}$$

Let $\mathbf{X}_t^p$ denote the predicted trajectory at time t, we then evaluate the tolerance $r_{time}(\mathbf{X}_t^p)$ and $r_{space}(\mathbf{X}_t^p)$ for the predicted trajectory. If at time t we have $r_{time}(\mathbf{X}_t^p) \geq \theta_{time}$ and $r_{space}(\mathbf{X}_t^p) \geq \theta_{space}$, then it indicates the execution sequence is able to satisfy the specification and there is no need for correction. We simply use $U_r(t)$ from offline calculation as the control inputs at time t. Otherwise, the event-trigger MPC module will be activated and correct the execution.

5.1 Event-triggered Model Predictive Control

An event-triggered MPC is designed for runtime self-correction, where we are constantly evaluating whether the predicted trajectory still satisfies the given specification and maintains a specific tolerance degree. If $r_{space}(\mathbf{X}_t^p) < \theta_{space}$ or $r_{time}(\mathbf{X}_t^p) < \theta_{time}$ at time t, it suggests possible violations for the specifications in the future and the MPC module will be triggered. The MPC problem is formulated as follows:

$$\begin{aligned} \min_{\mathbf{X}(t), u(t)} \quad & \sum_{\tau=t}^{\tau=t+T} (X_r(\tau) - \mathbf{X}(\tau))^T Q (X_r(\tau) - \mathbf{X}(\tau)) \\ \text{subject to} \quad & \mathbf{X}(\tau + 1) = f(\mathbf{X}(\tau), u(\tau)), \tau \in [t, t + T - 1] \\ & \mathbf{X}(t + T) = X_r(t + T) \end{aligned} \tag{17}$$

where T is the horizon. By solving the MPC problem, we try to bring the robot back to the reference trajectory. Note that only the first step of the computed optimal control strategy (denoted as $u^*(t)$) is implemented, i.e, at time t, we use $u^*(t)$ instead of $U_r(t)$ as the control input. We will re-evaluate the predicted states at the next time step iteratively until the end of the planning trajectory.

6. CASE STUDIES

In this section, we consider two different case studies, where the first one has tighter time constraints and the second one has tighter space constraints. The experiments are run through YALMIP-CPLEX on a computer with 2.8GHz processor and 8GB memory. The MPC has a horizon T=10. For both examples, we use $\theta_{space} = 0.3$ and $\theta_{time} = 4$.

Example 6.1. We first consider a sequential task that the robot needs to visit position A between 10 and 20 seconds, and visit B between 21 and 31 seconds, visit C between 32 and 42 seconds, and never be in the yellow regions O_is ($i \in [1, k_1]$, where k_1 is the number of obstacles). The STL specification is given as below.

$$\varphi_1 = \Diamond_{[10,20]} A \wedge \Diamond_{[21,31]} B \wedge \Diamond_{[32,42]} C \wedge (\bigwedge_{i=1,\cdots,k_1} \Box \neg O_i) \tag{18}$$

Region A is represented as $(x > 2 \wedge x < 3 \wedge y > 6 \wedge y < 7)$ and similarly for other regions. The optimization problem

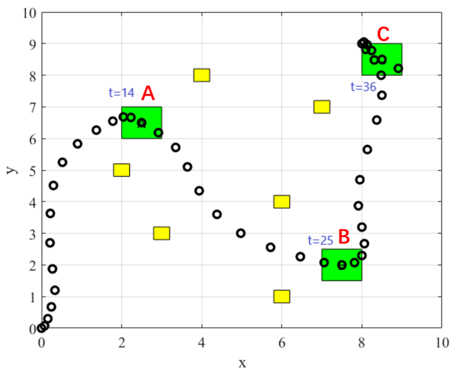

Fig. 2. Resulting path with maximum space and time tolerances for φ_1. The blue text shows the time that the robot enters each green region.

is formulated as in equation (1). We assume the velocity information is perfect when we are generating the reference trajectory offline, and is not perfect with a white noise deviation added during runtime execution. The resulting reference trajectory is shown in Fig 2. As can be seen from Fig 2, the final path of the robot stays far away from the yellow regions, and always goes through the center of the green regions for maximum space robustness. The robot also slows down when it enters green regions to maximize time robustness. Fig 3 shows that when the disturbance is small, the robot is able to still satisfy the specification without any correction.

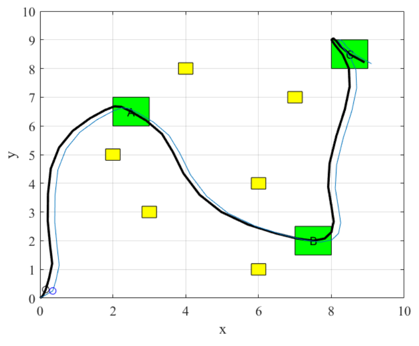

Fig. 3. Monitoring runtime sequence (blue line) with space and time tolerances. The monitor indicates that the runtime sequence also satisfies φ. No correction is needed and MPC never turns on.

However, when the deviation is large, the MPC module will be turned on and guide the robot to satisfy the desired specification with self-corrections. Note that the blue dashed line in Fig 4 is the predicted trajectory at $t = 8$, and it is not able to reach position C thus violating the specification. Fig 5 shows the triggering instances of MPC, and the MPC module has been triggered for 4 seconds in total in this example.

Example 6.2. In the second example, we consider an environment with more obstacles but with a relatively looser time constraints. The specification is given as below, where we require the robot to eventually visit position A between 10 seconds and 20 seconds, and eventually visit position C between 32 seconds and 42 seconds while avoiding all k_2 obstacles.

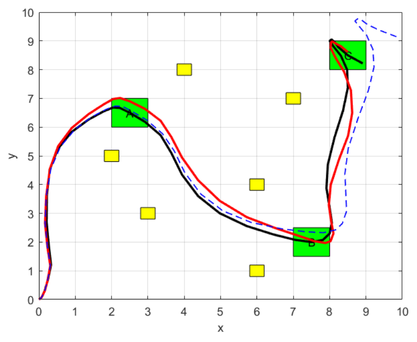

Fig. 4. Resulting trajectory for φ_1 with self-correction. The blue dashed line indicates the predicted path at $t = 8$. The red line shows the path with self-corrections. The reference trajectory is marked in black.

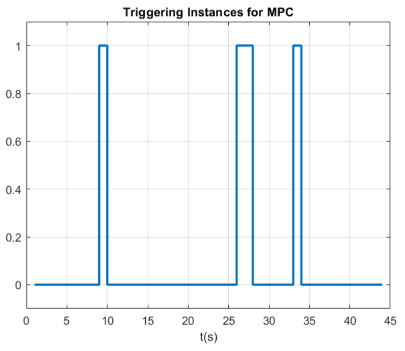

Fig. 5. Triggering instances for MPC. The MPC module has turned on for 4 seconds in total.

$$\varphi_2 = \Diamond_{[10,20]} A \wedge \Diamond_{[32,42]} C \wedge \big(\bigwedge_{i=1,\cdots,k_2} \Box \neg O_i \big) \tag{19}$$

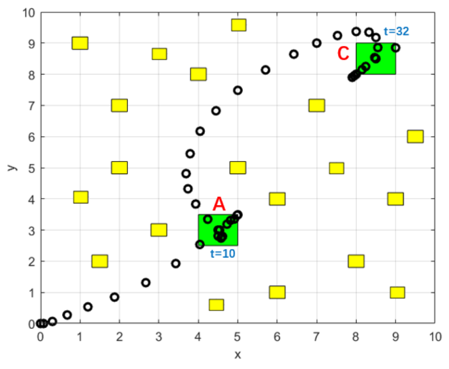

Fig. 6. Resulting path with maximum space and time tolerances for φ_2

Similarly, the offline planning is able to generate a path that maximize the space and time tolerances as shown in Fig 6. It is clearly visible in this case that the trajectory tends to stay in green regions as long as possible during the required time interval for maximum time robustness. During runtime execution, the blue dashed line in Fig 7 is the predicting trajectory at $t = 6$, and it reaches position C at the last time step. The time robustness requirement is thus violated and therefore MPC is triggered. Fig 8 shows the triggering instances of MPC, and MPC has been triggered for 11 seconds in total in this example.

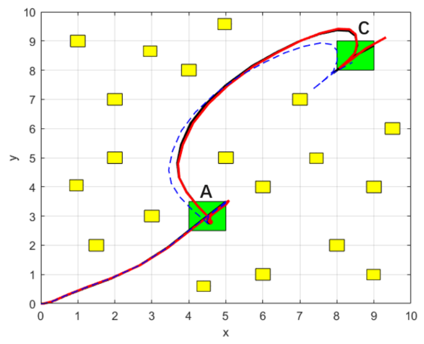

Fig. 7. Resulting trajectory for φ_2 with self-correction. The blue dashed line indicates the predicted path at $t = 6$. The red line shows the path with self-corrections.

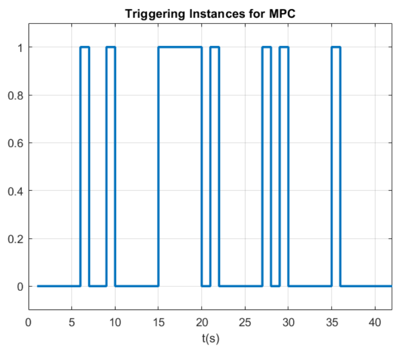

Fig. 8. Triggering instances for MPC. The MPC module has turned on for 11 seconds in total.

Compared to the first example, MPC has been triggered more frequently due to the complexity of the environment (It is more likely to hit an obstacle). Table 1 summarizes the number of linear constraints and computation time for each of the examples.

Table 1. Number of constraints and computation time

STL Specifications	# of linear constraints	Computation time (s)
φ_1	3154	10123
φ_2	2615	9673

7. CONCLUSIONS

In this paper, we have presented an optimization-based approach for robot planning, monitoring and self-correction problems under STL specifications with finite time constraints. Our approach translates the STL specifications into mixed-integer linear constraints, and the goal of the optimization problem is to maximize the overall space and time tolerances under double integrator dynamics of the robotic agent. During runtime execution, we consider a realistic situation where the velocity information is not perfect. A prediction module and a self-correction module with event-triggered model predictive control have been designed to predict and prevent possible future violations of the specifications. The simulation results show promising performance of our approach to find an optimal solution, and the robotic agent is able to make self-corrections during runtime execution when the velocity information is noisy.

Since we have used a binary variable (z) with each halfspace, the problem would be complex if the environment contains too many halfspaces. Therefore, the future directions of this work could include task decomposition and reduction of binary variables. Other aspects such as learning from the self-corrections, and multi-robot cooperative planning could also be possible extension of this work.

REFERENCES

Akazaki, T. and Ichiro, H. (2015). Time robustness in mtl and expressivity in hybrid system falsification. *International Conference on Computer Aided Verification.*

Alur, R., Feder, T., and Henzinger, T.A. (1996). The benefits of relaxing punctuality. *Journal of the ACM*, 43, 116–146.

Bhatia, A., Maly, M.R., Kavraki, L.E., and Vardi, M.Y. (2011). Motion planning with complex goals. *IEEE Robotics Automation Magazine.*

Donze, A. and Maler, O. (2010). Robust satisfaction of temporal logic over real-valued signals. *International Conference on Formal Modeling and Analysis of Timed Systems.*

Fainekos, G.E. and Pappas, G.J. (2009). Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science.*

Karaman, S., Sanfelice, R.G., and Frazzoli, E. (2008). Optimal control of mixed logical dynamical systems with linear temporal logic specifications. *IEEE Conference on Decision and Control.*

Lamport, L. (1994). The temporal logic of actions. *ACM Transactions on Programming Languages and Systems.*

Lin, Z. and Baras, J. (2019). Planning and runtime monitoring of robotic manipulator using metric interval temporal logic. *IEEE System Conference.*

Lindemann, L. and Dimarogonas, D.V. (2017). Robust motion planning employing signal temporal logic. *American Control Conference.*

Lindemann, L., Maity, D., Baras, J.S., and Dimarogonas, D.V. (2018). Event-triggered feedback control for signal temporal logic tasks. *IEEE Conference on Decision and Control.*

Plaku, E. and Karaman, S. (2016). Motion planning with temporal-logic specifications: Progress and challenges. *AI communication.*

Smith, S.L., Tumova, J., Belta, C., and Rus, D. (2010). Optimal path planning under temporal logic constraints. *IEEE/RSJ International Conference on Intelligent Robots and Systems.*

Vasumathi, R., Donze, A., Maasoumy, M., and Murray, R.M. (2014). Model predictive control with signal temporal logic specifications. *53rd IEEE Conference on Decision and Control.*

Wolff, E., Topcu, U., and Murray, R.M. (2014). Optimization-based trajectory generation with linear temporal logic specifications. *International Conference on Robotics and Automation.*

Zhou, Y., Maity, D., and Baras, J.S. (2015). Optimal mission planner with timed temporal logic constraints. *European Control Conference.*