IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Receding Horizon Control for Signal Temporal Logic using Robustness-Conserving Partial Formula Evaluation

Roland Ilyes, Lara Brudermüller, Nick Hawes, and Bruno Lacerda

***Abstract*—We present a bounded-memory receding horizon approach to robot control for complex specifications in dynamic environments. We use Signal Temporal Logic, a logic that quantifies how robustly trajectories satisfy the specification, to specify robot behavior. To handle unbounded specifications, we consider a short planning horizon, only searching for nonviolating trajectories. We identify the subset of Signal Temporal Logic for which this approach needs only a bounded memory of the past, and leverage syntactic separation to summarize the robust satisfaction of the trajectory as it evolves. We implement our approach using receding horizon control in dynamic environments. We demonstrate the effectiveness and scalability of our approach compared to the state-of-the-art approach in several case studies.**

***Index Terms*—Formal Methods in Robotics and Automation, Hybrid Logical/Dynamical Planning and Verification, Optimization and Optimal Control**

I. INTRODUCTION

ROBOTS operate in a broad scope of applications, and must often perform complex tasks. Temporal Logics [1] formalize such tasks as *specifications*. Signal Temporal Logic (STL) [2] describes tasks by expressing relations between events using real-valued time constraints. STL is distinct from other logics in that it expresses events as functions, called *predicates*, of the system state. These predicates allow algorithms to measure how much a trajectory satisfies or violates an STL specification. This measure is known as the *robust satisfaction value*, or *robustness* [3], and characterizes how robust a trajectory is to disturbances.

In this work, we are interested in leveraging STL for robot control in dynamic environments. Consider a practical example where a robot must periodically transport packages from a truck to a human. STL can express this task and also specify how frequently the robot must move between the two. However, such a specification reasons over an unbounded time horizon. Furthermore, the robot might be unaware of how the human will move and must react to the human's motion. One approach to achieve this reactive behavior over the unbounded task duration is receding horizon control (RHC). For a given time t, RHC uses the current state of the robot and environment to compute an optimal trajectory over some finite horizon $[t, t+h]$, and executes the first control input. Each trajectory assumes the environment is static, but the iterative replanning gives RHC *inherent* reactivity to dynamic environments [4].

The trajectory optimization problem can encode the STL specification either as integer constraints in a mixed integer convex program (MICP) [5] or as the objective function in a nonlinear program (NLP) [6]. These encodings reason over finite-horizon, or *bounded* specifications. However, the RHC framework can attempt to enforce the bounded specification to hold at every time t by maintaining a bounded memory of the past trajectory [5]. This technique allows RHC to iteratively produce trajectories that satisfy a small set of *unbounded* specifications.

In this work, we extend RHC for STL to a more general set of unbounded specifications than the approach in [5]. To accomplish this, we use the Robust Satisfaction Interval (RoSI), a runtime verification tool that monitors the robustness of trajectories as they evolve [7]. By incorporating this into an objective function, we formulate an NLP that searches for nonviolating trajectories over the planning horizon. The RoSI is defined over arbitrary STL, allowing this approach to generalize to unbounded specifications. However, maintaining the RoSI requires unbounded memory of the past in general, preventing RHC from operating indefinitely. The authors in [7] identify a subset of STL for which maintaining the RoSI requires only a bounded memory. In this work, we expand this subset using syntactic separation [8], a specification rewriting technique. We use syntactic separation to partially evaluate specifications and summarize the robustness of the past, expanding the subset of STL for which maintaining the RoSI requires a bounded memory.

The contributions of this paper are (i) a receding horizon problem formulation for STL, (ii) a robustness-conserving STL formula rewriting approach with bounded memory for a subset of STL, and (iii) equations for updating the rewritten formula online, for use in an RHC algorithm. Our approach combines the formula rewriting technique with RoSI optimization, enabling RHC for a new subset of unbounded STL.

II. RELATED WORK

The first work on control synthesis for STL was [5], where they extend an MICP approach to synthesis for Linear Temporal Logic (LTL) specifications from [9] to handle STL specifications. This approach enforces the specification using integer constraints in an optimal control problem. If the system is linear and the objective function is convex, the resulting problem is an MICP [10]. Robust extensions of this approach have been investigated in [11], [12], and methods to reduce

Manuscript received: April, 22, 2025; Revised July, 20, 2025; Accepted August, 19, 2025.

This paper was recommended for publication by Editor Lucia Pallottino upon evaluation of the Associate Editor and Reviewers' comments. This work received EPSRC funding via the "From Sensing to Collaboration" programme grant [EP/V000748/1]. R.I. and L.B. were supported by Amazon Web Services Lighthouse scholarships.

All authors are with the Oxford Robotics Institute, University of Oxford, United Kingdom. For correspondence: {rolandilyes, larab, nickh, bruno}@robots.ox.ac.uk

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

the number of generated constraints have been investigated in [13]. The MICP approach creates binary variables for every timestep, and this scales poorly for long-horizon specifications. Moreover, the MICP encodings only apply to STL specifications over linear predicates and for systems with linear dynamics.

An alternate approach uses the STL robustness measure as the objective function, and allows for nonlinear predicates and dynamics. Robustness is a nonlinear and discontinuous function, resulting in an NLP. The first application of optimization for temporal logic robustness was [14], where the authors minimize Metric Temporal Logic (MTL) robustness for verification purposes. MTL robustness is a highly discontinuous function, and so the authors in [15] propose a smooth approximation of MTL robustness for control synthesis using gradient ascent techniques. The approaches [16], [6], [17] extend gradient ascent to optimize smooth approximations of STL robustness, while [18], [19] define new "average" STL robustness measures to optimize for. Due to the lack of mixed integer constraints, this approach scales better than the MICP approach, making it more practical for RHC.

These approaches are typically incorporated in closed-loop control using a "shrinking horizon" approach, where each planning iteration considers the full time domain of the specification. At each iteration, the robot state and control inputs corresponding to prior timesteps are constrained to the values decided in earlier planning iterations. Shrinking horizon control requires that the specification is *bounded*, reasoning over a finite time horizon. For *unbounded* specifications, the authors in [5] propose a receding horizon approach for a subset of STL. In contrast, we propose a receding horizon approach that applies to a more general subset of STL that requires only that nested temporal operators are bounded.

Our approach to receding horizon control for STL leverages the Robust Satisfaction Interval (RoSI). This is a tool for runtime verification that bounds the achievable robustness of trajectories as they evolve [7]. The authors in [20] extend the RoSI to reason over the past, and use it to synthesize formulas to describe patterns in data. The RoSI has been used for sampling-based control synthesis in [21], where the authors incrementally find open-loop motion plans for bounded STL specifications. In contrast, we use the RoSI for closed-loop receding horizon control by optimizing the supremum of the RoSI at each timestep.

For RHC to operate indefinitely, it must require only a bounded memory of the past. The authors in [7] provide a set of STL specifications for which the RoSI can be maintained with a bounded memory by *summarizing* robustness values of the past. In this work, we extend the set of STL specifications for which this is possible and generalize the result from [7]. To this end, we use *syntactic separation*, an operation that splits a specification into multiple specifications over separate time domains [8]. Syntactic separation results in an overlap of the resulting time domains, unless the formula does not nest temporal operators. [22] investigates such formulas, leveraging syntactic separation to propose efficient shrinking horizon control by planning for smaller time domains iteratively. Similarly, [23] applies syntactic separation to a larger subset of STL for iterative control synthesis. They propose a restriction to resolve the overlap of time domains, where the restricted specification describes a subset of the behaviors described by the original specification. In contrast to these approaches, we use syntactic separation to identify the past portion of STL formulas for which the robustness can be summarized, and keep the overlap in time domains as a bounded memory in our RHC approach.

III. PRELIMINARIES

Let $\boldsymbol{x} \in X \subseteq \mathbb{R}^n$ be a *state*, and time domain $\mathcal{T}$ be a set of time instants such that $\mathcal{T} \subseteq \mathbb{R}_{\geq 0}$. Then, a *signal* s is a mapping from $\mathcal{T}$ to X. A *signal predicate* μ is a formula of the form $g(\boldsymbol{x}) > 0$, where $g : X \to \mathbb{R}$. We express properties of signals with respect to signal predicates using *signal temporal logic* (STL) [2].

Definition 1: The STL Syntax is recursively defined by:

$$\varphi := \top \mid \mu \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi U_I \varphi$$

where φ are STL specifications, also called *formulas*, and I = $\langle a, b\rangle$ is a time interval that can be open or closed at both ends, with $a, b \in \mathbb{R}$. Notation $\top$, $\neg$, $\wedge$ are Boolean "true", "negation", and "conjunction," respectively, and U denotes the temporal "until" operator.

These connectives can define other operations such as (i) Boolean disjunction $\varphi \vee \psi = \neg(\neg\varphi \wedge \neg\psi)$, (ii) Boolean implication $\varphi \to \psi = \neg\varphi \vee \psi$, (iii) the "eventually" operator $\Diamond_I \varphi = \top U_I \varphi$, and (iv) the "globally" operator $\Box_I \varphi = \neg(\Diamond_I \neg\varphi)$.

We treat intervals as sets, where $\cup$ is the union of sets, $\cap$ is the intersection, $I_1 + I_2 = \{i_1 + i_2 \mid i_1 \in I_1, i_2 \in I_2\}$ is the Minkowski sum, and $\langle a, b\rangle + t = \langle a + t, b + t\rangle$.

Definition 2: The semantics of STL is defined over a signal $s : \mathcal{T} \to X$ at time t as:

$$\begin{aligned}
(s,t) \vDash \top \quad &\iff \top \\
(s,t) \vDash g(\boldsymbol{x}) > 0 \quad &\iff g(s(t)) > 0 \\
(s,t) \vDash \neg\varphi \quad &\iff (s,t) \nvDash \varphi \\
(s,t) \vDash \varphi \wedge \psi \quad &\iff (s,t) \vDash \varphi \wedge (s,t) \vDash \psi \\
(s,t) \vDash \varphi U_I \psi \quad &\iff \exists t' \in ((I + t) \cap \mathcal{T}) \; s.t. (s,t') \vDash \psi \\
&\qquad \wedge \forall t'' \in ([t, t') \cap \mathcal{T}), (s,t'') \vDash \varphi
\end{aligned}$$

where $\vDash$ denotes satisfaction. A signal s satisfies an STL formula, denoted by $s \vDash \varphi$, if $(s, 0) \vDash \varphi$.

The inclusion of $\mathcal{T}$ above is to allow for both continuous and pointwise semantics. Refer to [24] and [25] for a discussion on the differences between continuous and pointwise semantics.

Definition 3: The time domain of an STL formula $[\varphi] = [[\varphi]^{\downarrow}, [\varphi]^{\uparrow}]$ is:

$$\begin{aligned}
[\top] \quad &= \emptyset \\
[g(\boldsymbol{x}) > 0] \quad &= [0, 0] \\
[\neg\varphi] \quad &= [\varphi] \\
[\varphi \wedge \psi] \quad &= [\varphi] \cup [\psi] \\
[\varphi U_{\langle a,b\rangle} \psi] \quad &= ([\varphi] + [0, b)) \cup ([\psi] + \langle a, b\rangle).
\end{aligned}$$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

The time domain is the interval within which a signal state $\{s(t') \mid t' \in [\varphi]\}$ can impact the satisfaction of $s \vDash \varphi$. A formula is bounded if $[\varphi]^\uparrow < \infty$ or unbounded otherwise.

In addition to qualitative semantics, STL also admits *quantitative semantics*, known as the *robust satisfaction value*, or simply *robustness* [3].

Definition 4: The quantitative semantics, or robust satisfaction value, of STL formula φ is defined over a signal s at time t as $\rho(s,t,\varphi)$:

$$\begin{aligned}
\rho(s,t,\top) &= \infty \\
\rho(s,t,g(\boldsymbol{x})>0) &= g(s(t)) \\
\rho(s,t,\neg\varphi) &= -\rho(s,t,\varphi) \\
\rho(s,t,\varphi\wedge\psi) &= \min\{\rho(s,t,\varphi),\rho(s,t,\psi)\} \\
\rho(s,t,\varphi U_I\psi) &= \sup_{t'\in(I\cap\mathcal{T})} \min\begin{pmatrix}\rho(s,t',\psi),\\ \inf_{t''\in([t,t')\cap\mathcal{T})}\rho(s,t'',\varphi)\end{pmatrix}.
\end{aligned}$$

A signal s satisfies an STL formula φ if $\rho(s,0,\varphi)>0$.

Typically, STL syntax and semantics do not allow for negative-length intervals, where $a<b$. In this work, however, we allow them and treat them as the empty set $\emptyset$. By Def. 2, $(s,t) \vDash \varphi U_\emptyset \psi \iff \bot$, because there exists no $t' \in \emptyset$. Similarly, by Def. 4, $\rho(s,t,\varphi U_\emptyset\psi) = -\infty$, because the supremum of an empty set is $-\infty$.

Let $\tilde{t} \in \mathbb{R}_{\geq 0}$ and $s_{\tilde{t}} : [0,\tilde{t}] \to X$. Then, $s_{\tilde{t}}$ is a *prefix* of signal s if, for all $t' \leq \tilde{t}, s(t') = s_{\tilde{t}}(t')$. Let $C(s_{\tilde{t}}) = \{s \mid s_{\tilde{t}} \text{ is a prefix of } s\}$ be the set of *completions* of $s_{\tilde{t}}$.

The authors in [7] introduce interval-based semantics to bound the robustness of the completions of a given prefix.

Definition 5: The Robust Satisfaction Interval (RoSI) of an STL formula φ on a partial signal $s_{\tilde{t}}$ at time $t \in \mathbb{R}_{\geq 0}$ is an interval $[\rho](s_{\tilde{t}},t,\varphi)$ such that:

$$\begin{aligned}
\inf([\rho](s_{\tilde{t}},t,\varphi)) &= \inf_{s\in C(s_{\tilde{t}})}\rho(s,t,\varphi) \\
\sup([\rho](s_{\tilde{t}},t,\varphi)) &= \sup_{s\in C(s_{\tilde{t}})}\rho(s,t,\varphi).
\end{aligned}$$

The authors in [7] provide a recursive function to calculate $[\rho](s_{\tilde{t}},t,\varphi)$. In this paper, we use the notation $\inf([\rho](s_{\tilde{t}},t,\varphi)) = [\rho]^\downarrow(s_{\tilde{t}},t,\varphi)$ and $\sup([\rho](s_{\tilde{t}},t,\varphi)) = [\rho]^\uparrow(s_{\tilde{t}},t,\varphi)$.

IV. PROBLEM FORMULATION

Consider a continuous time robotic system of the form $\dot{\boldsymbol{x}}_t^r = f(\boldsymbol{x}_t^r,\boldsymbol{u}_t)$, where $\boldsymbol{x}_t^r \in X^r \subset \mathbb{R}^{n_r}$ is the state of the robot and $\boldsymbol{u}_t \in U \subset \mathbb{R}^m$ is the control input. Given a sampling time $\Delta > 0$, we assume that the system admits a discrete time approximation of the form $\boldsymbol{x}_{t+\Delta}^r = \tilde{f}(\boldsymbol{x}_t^r,\boldsymbol{u}_t)$.

The environment state is $\boldsymbol{x}_t^e \in X^e \subset \mathbb{R}^{n_e}$, and its dynamics are unknown. The environment can capture dynamic yet not controllable aspects of the world that are relevant to the task. This includes not only a dynamic physical space, but also other agents acting in the same space. Note that the environment state refers only to dynamic components of the environment, while static components of the environment are described using traditional signal predicates.

We wish to specify tasks as STL formulas over the composed robot-environment system. For a specification φ, we consider time domain $\mathcal{T} = \{0,\Delta,2\Delta,...,[\varphi]^\uparrow\}$. Note that, if φ is unbounded, the set $\mathcal{T}$ is infinite. Our signal $s \in S$ is a mapping $s : \mathcal{T} \to X^r \times X^e \times U$, and prefix signals $s_{\tilde{t}} \in S_{\tilde{t}}$ are mappings $s_{\tilde{t}} : [0,\tilde{t}] \cap \mathcal{T} \to X^r \times X^e \times U$. The inclusion of U allows for expressing constraints on the control input. We extract components of the signal using superscripts, i.e., the robot state component of signal s at time t is $s^r(t)$.

Problem 1: Given an STL specification φ, initial robot state $\boldsymbol{x}_0^r$, initial environment state $\boldsymbol{x}_0^e$, and system $\tilde{f}$, consider signal s where $s^r(0) = \boldsymbol{x}_0^r$, $s^e(0) = \boldsymbol{x}_0^e$, and $s^r(t+\Delta) = \tilde{f}(s^r(t),s^u(t))$ for all $t \in [0,[\varphi]^\uparrow - \Delta]$. Find a control input sequence s^u such that $(s,0) \vDash \varphi$.

Due to the uncontrollable environment signal with unknown dynamics s^e, no open-loop control signal s^u is guaranteed to solve Problem 1. Instead, closed-loop approaches build s^u incrementally online while observing the environment signal s^e. Such approaches provide soundness guarantees, meaning that any solutions they find are indeed solutions.

For bounded specifications, the approach to solving Problem 1 is to optimize trajectories in a *shrinking horizon* manner. This approach plans over the entire formula horizon $[[\varphi]^\downarrow,[\varphi]^\uparrow]$ at each timestep. For a given timestep t, the shrinking horizon approach constrains the solution from $[[\varphi]^\downarrow,t]$ to the values decided in prior planning iterations, and assumes a static environment over $[t,[\varphi]^\uparrow]$. At each planning iteration, MICP approaches encode the specification as integer constraints on the trajectory, while NLP approaches search for trajectories that optimize the robustness.

For unbounded specifications, the authors in [5] propose a *receding horizon* approach for unbounded specifications in the set $\Phi_{\text{Uni-Temporal}} = \{\varphi U_{[0,\infty]}\psi, \square_{[0,\infty]}\varphi, \Diamond_{[0,\infty]}\varphi\}$, where φ and ψ are bounded formulas. For brevity, we describe the approach for $\square_{[0,\infty]}\varphi$, called *safety* specifications. At each planning iteration t, this approach plans to satisfy φ over $[t,t+[\varphi]^\uparrow]$. To ensure satisfaction of $\square_{[0,\infty]}\varphi$, this approach must also ensure satisfaction at previous instants $(s,t') \vDash \varphi$, for all $t' \in [t-[\varphi]^\uparrow,t)$. The satisfaction of points prior to $t-[\varphi]^\uparrow$ has already been decided, allowing the receding horizon approach to maintain only a bounded memory of the past. Such bounded memory is integral for allowing the approach to operate indefinitely.

In this work, we aim to expand the receding horizon approach beyond specifications in $\Phi_{\text{Uni-Temporal}}$. To this end, our first contribution is to optimize the supremum of the RoSI at each planning instance, resulting in an NLP. The RoSI is defined over arbitrary STL, allowing the RHC approach to generalize beyond $\Phi_{\text{Uni-Temporal}}$.

Problem 2: Given STL specification φ, system $\tilde{f}$, prefix signal $s_{\tilde{t}}$, and planning horizon h, find

$\arg\max_{s_{\tilde{t}+h}\in S_{\tilde{t}+h}} [\rho]^\uparrow(s_{\tilde{t}+h},0,\varphi)$ such that:

(i) $s_{\tilde{t}+h}(t') = s_{\tilde{t}}(t') \forall t' \leq \tilde{t}$,

(ii) $s^e_{\tilde{t}+h}(t') = s^e_{\tilde{t}}(t') \forall t' \geq \tilde{t}$, and

(iii) $s^r_{\tilde{t}+h}(t'+\Delta) = \tilde{f}(s^r_{\tilde{t}+h}(t'),s^u_{\tilde{t}+h}(t')) \forall t' \in [0,\tilde{t}+h-\Delta]$.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Here, (i) ensures that the solution contains the observed history of the robot trajectory, (ii) assumes that the environment is static for planning purposes, and (iii) ensures that the planned state and control components of the signal obey the robot dynamics $\tilde{f}$. At each planning iteration, the RHC approach sends the control input $s^u_{t+h}(\tilde{t})$ to the robot.

V. ROBUSTNESS-CONSERVING PARTIALLY EVALUATED SIGNAL TEMPORAL LOGIC FORMULAS

Although the RoSI is defined over arbitrary STL, maintaining the RoSI requires an unbounded memory in general. With unbounded memory requirements, the RHC approach cannot operate indefinitely. The rest of this section discusses how to bound this memory. Intuitively, at each planning step, we wish to *separate* the part of the formula that reasons over the prefix signal, summarize the robustness contribution of that part, and plan for the remaining formula. We begin with an example of such a technique before formalizing it.

Example 1 (Robustness Summarization): Consider the formula $\varphi = \square_{[0,10]}(x > 3)$ and consider a constant prefix signal s_5 such that $s^x_5(t) = 3.5 \; \forall \; t \in [0, 5]$. The robustness of a signal with regards to φ is a function over all $t \in [\varphi] = [0, 10]$, and we want to reduce the size of the time domain under consideration. Because we already have data in $t = [0, 5]$, we recognize that we can *partially evaluate* the robustness:

$$\begin{aligned}\rho(s, 0, \varphi) &= \min_{t\in[0,10]}(s^x(t) - 3)\\ &= \min\{\min_{t\in[0,5]}(s^x_5(t) - 3), \min_{t\in[5,10]}(s^x(t) - 3)\}\\ &= \min\{0.5, \min_{t\in[5,10]}(s^x(t) - 3)\}.\end{aligned}$$

This function happens to also be the robustness of the formula $\varphi' = (0.5 > 0) \wedge \square_{[5,10]} x > 3$. This new formula only reasons over $t = [5, 10]$, but results in the same robustness function. Intuitively, we have *summarized* the robustness of the signal up until $t = 5$ in a new *induced predicate* $(0.5 > 0)$. With this new formula φ', we can evaluate the robustness of the original formula φ on the entire trajectory s while only considering the time domain $t = [5, 10]$.

To formalize the portion of the formula that can be summarized, we use *syntactic separation* [8]. Syntactic separation rewrites MTL formulas as Boolean combinations of formulas with different time domains, while maintaining the semantic equivalence. STL is a subset of MTL, meaning syntactic separation is directly applicable to STL.

Definition 6: For $\bar{t} \geq 0$,

$$\text{sep}(\varphi U_{\langle a,b\rangle}\psi, \bar{t}) = \varphi U_{\overline{I}_1}\psi \vee (\square_{\overline{I}_2}\varphi \wedge \Diamond_{[\bar{t},\bar{t}]}(\varphi U_{\overline{I}_3}\psi)),$$

where

$$\begin{aligned}\overline{I}_1 &= I_1(\bar{t}, \langle a, b\rangle) = (-\infty, \bar{t}] \cap \langle a, b\rangle\\ \overline{I}_2 &= I_2(\bar{t}, \langle a, b\rangle) = (-\infty, \bar{t}] \cap [0, b\rangle\\ \overline{I}_3 &= I_3(\bar{t}, \langle a, b\rangle) = [0, \infty) \cap \langle a - \bar{t}, b - \bar{t}\rangle.\end{aligned}$$

Here, the intervals I_1, I_2, I_3 utilize the reasoning over empty sets discussed in Section III to apply to any $\bar{t} \geq 0$. Syntactic separation aims to split the formula into subformulas that reason over different time domains, and the rewritten formula can be interpreted as $(\texttt{already satisfied} \vee (\texttt{has not violated} \wedge \texttt{will be satisfied}))$. The partial evaluation from Example 1 isolates the part of the formula prescribing properties over the past, that is, subformulas `already satisfied` and `has not violated`. However, these subformulas prescribe properties over time *beyond* the splitting point $\bar{t}$. From Def. 3, the supremum of the union of the time domain of these subformulas is $([\texttt{already satisfied}] \cup [\texttt{has not violated}])^\uparrow = \min\{\bar{t}, b\} + \max\{[\varphi]^\uparrow, [\psi]^\uparrow\}$. When $\max\{[\varphi]^\uparrow, [\psi]^\uparrow\} > 0$, the supremum of this time domain is greater than $\bar{t}$. To make the time domain end at or before a point $\hat{t}$, we must place the splitting point earlier, at

$$\bar{t} = \hat{t} - \max\{[\varphi]^\uparrow, [\psi]^\uparrow\}.$$

The early splitting point is a manifestation of the *memory* of an STL formula.

Definition 7: The memory of an STL formula φ is defined recursively as:

$$m(\varphi) = \begin{cases} 0 & \text{if } \varphi \in \{g(\boldsymbol{x}) > 0, \top\}\\ m(\varphi) & \text{if } \varphi = \neg\varphi\\ \max\{m(\varphi), m(\psi)\} & \text{if } \varphi = \varphi \wedge \psi\\ \max\{[\varphi]^\uparrow, [\psi]^\uparrow\} & \text{if } \varphi = \varphi U_{\langle a,b\rangle}\psi,\end{cases}$$

where $[\varphi]^\uparrow$ is defined in Def. 3.

For a prefix signal $s_{\tilde{t}}$, bounded-memory STL formula $m(\varphi) \leq \infty$, and splitting point $\bar{t} = \tilde{t} - m(\varphi)$, syntactic separation results in subformulas that reason over a domain within which the signal is fully defined. This enables generalization of the robustness summarization presented in Example 1: Partial evaluation of the syntactically separated formula *defines* induced predicates which summarize the contribution of the prefix signal to the robustness, resulting in a formula with a smaller time domain.

Definition 8: The Robustness-Conserving Partially Evaluated (RCPE) formula is defined over a bounded-memory STL formula φ and a partial signal $s_{\tilde{t}}$ at time t, where $\tilde{t} \geq t + m(\varphi)$, by function $\text{RCPE}(s_{\tilde{t}}, t, \varphi)$:

$$\begin{aligned}&\text{RCPE}(s_{\tilde{t}}, t, \top) = \top\\ &\text{RCPE}(s_{\tilde{t}}, t, g(\boldsymbol{x}) > 0) = g(s_{\tilde{t}}(t)) > 0\\ &\text{RCPE}(s_{\tilde{t}}, t, \neg\varphi) = \neg\text{RCPE}(s_{\tilde{t}}, t, \varphi)\\ &\text{RCPE}(s_{\tilde{t}}, t, \varphi \wedge \psi) = \text{RCPE}(s_{\tilde{t}}, t, \varphi) \wedge \text{RCPE}(s_{\tilde{t}}, t, \psi),\\ &\text{RCPE}(s_{\tilde{t}}, t, \varphi U_{\langle a,b\rangle}\psi) = \overline{\mu}^U_{\tilde{t}} \vee \left(\overline{\mu}^{\square}_{\tilde{t}} \wedge \Diamond_{[\bar{t},\bar{t}]}(\varphi U_{I_3(\bar{t},\langle a,b\rangle)}\psi)\right)\end{aligned}$$

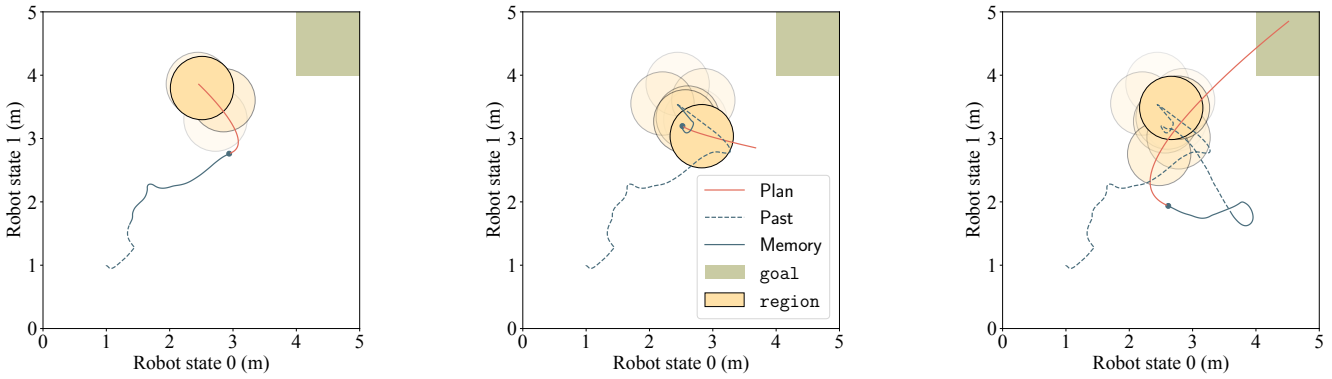
where

$$\begin{aligned}\overline{\mu}^U_{\tilde{t}} &= \rho(s_{\tilde{t}}, t, \varphi U_{I_1(\bar{t},\langle a,b\rangle)}\psi) > 0,\\ \overline{\mu}^{\square}_{\tilde{t}} &= \rho(s_{\tilde{t}}, t, \square_{I_2(\bar{t},\langle a,b\rangle)}\varphi) > 0,\end{aligned}$$

$\bar{t} = \tilde{t} - t - m(\varphi U_{\langle a,b\rangle}\psi)$, and I_1, I_2, I_3 are in Def. 6.

RCPE uses the memory of the STL formula and length of the signal prefix to pick a syntactic separation point such that the prefix is enough to evaluate the robustness of some of the resulting subformulas. This robustness is captured in the new signal predicates $\overline{\mu}^U_{\tilde{t}}$ and $\overline{\mu}^{\square}_{\tilde{t}}$ for the separated temporal operators, and $g(s_{\tilde{t}}(t)) > 0$ for predicates. The signal

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.



(a) Snapshot at $\tilde{t} = 5$. $RCPE(\varphi_{stayIn}) = ((-5.33 > 0) \vee (\Diamond_{[2,10]}\Box_{[0,3]}\texttt{region})) \wedge ((-\infty > 0) \vee (\Diamond_{[15,20]}\texttt{goal}))$

(b) Snapshot at $\tilde{t} = 10$. $RCPE(\varphi_{stayIn}) = ((0.05 > 0) \vee (\Diamond_{[7,10]}\Box_{[0,3]}\texttt{region})) \wedge ((-\infty > 0) \vee (\Diamond_{[15,20]}\texttt{goal}))$

(c) Snapshot at $\tilde{t} = 15$. $RCPE(\varphi_{stayIn}) = (0.09 > 0) \wedge ((-\infty > 0) \vee (\Diamond_{[15,20]}\texttt{goal}))$

Fig. 1: Snapshots of mission execution for $\varphi_{stayIn} = (\Diamond_{[0,10]}(\Box_{[0,3]}\texttt{region})) \wedge \Diamond_{[15,20]}\texttt{goal}$, where $\texttt{region} = (\boldsymbol{x}^r[0] - \boldsymbol{x}^e[0])^2 + (\boldsymbol{x}^r[1] - \boldsymbol{x}^e[1])^2 < 0.25$ and $\texttt{goal} = \boldsymbol{x}^r[0] > 4 \wedge \boldsymbol{x}^r[1] > 4$. The RCPE formulas are given below each snapshot. Prior environment positions are captured in lower opacity.

predicates $\overline{\mu}_t^U$ and $\overline{\mu}_t^{\Box}$ capture the robustness of subformulas $\texttt{already satisfied}$ and $\texttt{has not violated}$ from syntactic separation, respectively. These induced signal predicates are the mechanism by which the RCPE formula conserves the robustness of the original formula.

Theorem 1: For any bounded-memory STL formula φ, partial signal $s_{\tilde{t}}$, and time t, where $\tilde{t} \geq t + m(\varphi)$,

$$[\rho](s_{\tilde{t}}, t, \text{RCPE}(s_{\tilde{t}}, t, \varphi)) = [\rho](s_{\tilde{t}}, t, \varphi).$$

Proof: We provide a proof sketch. This result holds by induction on the structure of STL. The base case of $\varphi = \top$ is trivially true. For the base case of $\varphi = g(\mathbf{x}) > 0$, RCPE replaces this formula with a new predicate $g(s_{\tilde{t}}(t)) > 0$, where $g(s_{\tilde{t}}(t))$ is a constant that stores the value of the original predicate, meaning the RoSI of the two are equivalent by design. The until operator is a base case because $\text{RCPE}(s_{\tilde{t}}, t, \varphi U_{\langle a,b \rangle} \psi)$ does not apply the RCPE operation to φ or ψ. In this case, Theorem 1 relies on the fact that syntactic separation (Def. 6) conserves the RoSI according to Def. 5 from [7]. In other words, $[\rho](s_{\tilde{t}}, t, \varphi U_{\langle a,b \rangle} \psi) = [\rho](s_{\tilde{t}}, t, \text{sep}(\varphi U_{\langle a,b \rangle} \psi, \bar{t}))$. Furthermore, the RoSI converges to robustness when $\tilde{t} \geq [\varphi]^{\uparrow} + t$. Because $\bar{t} = \tilde{t} - t - m(\varphi U_{\langle a,b \rangle} \psi)$, the RoSI of the subformulas considered in $\overline{\mu}_t^U$ and $\overline{\mu}_t^{\Box}$ have converged on the robustness, allowing the summarization using induced signal predicates (similar to the predicate base case). The inductive steps for negation and conjunction are straightforward, as the RCPE operation distributes to the subformulas of these operations. ■

The formalization of bounded-memory STL and introduction of the RCPE formula is contribution (ii) from Section I. The authors in [7] prove that the RoSI requires only a bounded memory for a set of STL formulas. Their proof relies on robustness summarization, and each formula is handled independently. The notion of RCPE formulas generalizes their approach to all bounded-memory STL formulas.

Example 2 (RCPE Formulas): Consider the formula φ_{stayIn}, shown in Fig. 1. This formula says "within the next 10 seconds, the robot must enter and *stay* in the dynamic region for 3 seconds. Also, the robot must reach the goal between 15 and 20 seconds." Fig. 1 shows the RCPE formula for different prefix trajectories, all evaluated with respect to the initial time $t = 0$. Fig. 1a shows a prefix trajectory and resulting RCPE formula for prefix s_5. One of the resulting subformulas from syntactic separation at time $\bar{t} = 5 - m(\varphi_{stayIn}) = 5 - 3 = 2$ is $\Diamond_{[0,2]}(\Box_{[0,3]}\texttt{region})$, and this is the formula which can be evaluated using the prefix. The robustness of this formula on the prefix $\rho(s_5, 0, \Diamond_{[0,2]}(\Box_{[0,3]}\texttt{region})) = -5.33$, and this is captured in the new induced predicate $(-5.33 > 0)$. The induced predicate $(-\infty > 0)$ is a result of $\text{RCPE}(s_5, 0, \Diamond_{[15,20]}\texttt{goal})$ using the supremum over an empty set, which is $-\infty$ as discussed in Section III.

The negative value in the predicate $(-5.33 > 0)$ means that the robot has not satisfied the original subformula $\Diamond_{[0,10]}(\Box_{[0,3]}\texttt{region})$. Compare this with Fig. 1b, showing the RCPE formula for the prefix s_{10}. In contrast to Fig. 1a, s_{10} has satisfied the original subformula, resulting in a positive value for the induced predicate $(0.05 > 0)$.

The robot still has time to achieve a higher robustness value, and this is reflected in the RCPE subformula $\Diamond_{[7,10]}(\Box_{[0,3]}\texttt{region})$ in Fig. 1b. Compare this to Fig. 1c, showing the RCPE formula for the prefix s_{15}. The robot indeed went on to achieve a higher robustness value of 0.09, reflected in the updated induced predicate $(0.09 > 0)$. Unlike in Fig. 1b, however, no new data can impact the robustness of this subformula. This is reflected in the final RCPE formula, which now only reasons about the other subformula $\Diamond_{[15,20]}\texttt{goal}$.

VI. ITERATIVE PARTIAL EVALUATION FOR RECEDING HORIZON CONTROL

Our approach involves finding the RCPE formula at each planning step. Because the past cannot change, the prefix at each planning step includes the prefixes from prior iterations

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

and creating the new induced predicates $\overline{\mu}_{\bar{t}}^{U}$ and $\overline{\mu}_{\bar{t}}^{\square}$ involves redundant computations. However, we can use the previous RCPE formula to update the new one:

Theorem 2: Let s_{t_1} be a prefix of s_{t_2}, where $t_1 \geq 0$. Then, for any given $\varphi U_{\langle a,b\rangle}\psi$,

$$\overline{\mu}_{t_2}^{U} = \rho(s_{t_2}, t, \overline{\mu}_{t_1}^{U} \vee (\overline{\mu}_{t_1}^{\square} \wedge \Diamond_{[\bar{t}_1,\bar{t}_1]}\varphi U_{\overline{I}_1'}\psi)) > 0,$$
$$\overline{\mu}_{t_2}^{\square} = \rho(s_{t_2}, t, \overline{\mu}_{t_1}^{\square} \wedge \square_{\overline{I}_2'}\varphi) > 0,$$

where

$$\bar{t}_i = t_i - t - m(\varphi U_{\langle a,b\rangle}\psi),$$
$$\overline{I}_1' = [0, \bar{t}_2 - \bar{t}_1] \cap \langle a - \bar{t}_1, b - \bar{t}_1\rangle,$$
$$\overline{I}_2' = [\bar{t}_1, \bar{t}_2] \cap [0, b\rangle,$$

and $\overline{\mu}_t^U$, $\overline{\mu}_t^{\square}$ are in Def. 8.

Proof: We provide a proof sketch. Consider $\overline{\mu}_{t_2}^{U}$ and $\overline{\mu}_{t_2}^{\square}$ as defined in Def. 8: $\overline{\mu}_{t_2}^{U} = \rho(s_{t_2}, t, \varphi U_{I_1(\bar{t}_2,\langle a,b\rangle)}\psi) > 0$, $\overline{\mu}_{t_2}^{\square} = \rho(s_{t_2}, t, \square_{I_2(\bar{t}_2,\langle a,b\rangle)}\varphi) > 0$. Applying RCPE to these formulas for prefix s_{t_1} results in

$$\text{RCPE}(s_{t_1}, t, \varphi U_{I_1(\bar{t}_2,\langle a,b\rangle)}\psi) = \overline{\mu}_{t_1}^{U} \vee (\overline{\mu}_{t_1}^{\square} \wedge \Diamond_{[\bar{t}_1,\bar{t}_1]}\varphi U_{\overline{I}_1'}\psi),$$
$$\text{RCPE}(s_{t_1}, t, \square_{I_2(\bar{t}_2,\langle a,b\rangle)}\varphi) = \overline{\mu}_{t_1}^{\square} \wedge \square_{\overline{I}_2'}\varphi,$$

corresponding to the equations for $\overline{\mu}_{t_2}^{U}$ and $\overline{\mu}_{t_2}^{\square}$ above. ∎

Algorithm 1 gives an overview of our RHC approach to solving Problem 1 for bounded-memory STL formulas. The algorithm uses a circular buffer to store the signal prefix (line 1), the size of which is determined by the sampling time Δ and formula memory $m(\varphi)$. At each planning iteration (line 6), the algorithm uses the most recent RCPE formula φ and the bounded prefix to solve Problem 2 (line 7). The algorithm then updates the prefix (line 9) and the RCPE formula using Theorem 2 (line 10). The algorithm then sends the control input $s^u(t)$ from the solution to the robot as an input (line 15), observes the new environment state (line 16), and repeats the process. The algorithm tracks the RoSI of the prefix (line 11), and terminates if the plan is violating or satisfying, returning the result (lines 12-14).

Theorem 3 (Soundness): Algorithm 1 is sound with respect to Problem 1.

Proof: We provide a proof sketch. Algorithm 1 only returns `success` if the infimum of $[\rho](s_{t-\Delta}, 0, \varphi) > 0$. From Def. 5, this means that all completions of the executed prefix have a robustness greater than zero and, from Def. 4, this means that all completions satisfy the specification. ∎

The time domain of $\text{RCPE}(s_{\tilde{t}}, 0, \varphi)$ when $\tilde{t} \geq m(\varphi)$ is $[\tilde{t} - m(\varphi), [\varphi]^{\uparrow}]$. For formulas with bounded memory $m(\varphi) \leq \infty$ and sampling time $\Delta > 0$, this means that the RoSI $[\rho](s_{\tilde{t}+h}, 0, \text{RCPE}(s_{\tilde{t}}, 0, \varphi))$ requires only a bounded memory of the past over time domain $[\tilde{t} - m(\varphi), \tilde{t}]$. This allows Algorithm 1 to operate indefinitely for unbounded specifications.

Example 3 (RCPE in RHC): Consider again the formula φ_{stayIn} and planning snapshots in Fig. 1. For all snapshots, the solid line shows the memory required to evaluate robustness of the RCPE formula, and the dashed line shows data that no longer needs to be processed to evaluate the robustness. The solid line always contains the previous $m(\varphi_{stayIn}) = 3$ seconds of data and does not grow during mission execution.

Algorithm 1 Control$(\varphi_0, \boldsymbol{x}_0^r, \boldsymbol{x}_0^e, h, \Delta)$

1: $s_{t-\Delta} \leftarrow$ `circularBuffer`$(m(\varphi_0)/\Delta)$
2: $\varphi \leftarrow \varphi_0$
3: $\boldsymbol{x}^r \leftarrow \boldsymbol{x}_0^r$
4: $\boldsymbol{x}^e \leftarrow \boldsymbol{x}_0^e$
5: $t \leftarrow 0$
6: **while** `True` **do**
7: $\quad s_{t+h} \leftarrow$ `plan`$(\varphi, s_{t-\Delta}, \boldsymbol{x}^e, \boldsymbol{x}^r, h)$ ▷ Solve Problem 2
8: $\quad \boldsymbol{u} \leftarrow s_{t+h}^u(t)$
9: $\quad s_{t-\Delta}$`.append`$((\boldsymbol{x}^r, \boldsymbol{x}^e, \boldsymbol{u}))$
10: $\quad \varphi \leftarrow$ `update`$(\varphi, \boldsymbol{x}^r, \boldsymbol{x}^e, \boldsymbol{u})$ ▷ Theorem 2
11: $\quad [\rho] \leftarrow [\rho](s_{t-\Delta}, 0, \varphi)$
12: $\quad$ **if** $[\rho]^{\uparrow} \leq 0$ **then return** `fail`
13: $\quad$ **else if** $[\rho]^{\downarrow} > 0$ **then return** `success`
14: $\quad$ **end if**
15: $\quad \boldsymbol{x}^r \leftarrow \tilde{f}(\boldsymbol{x}^r, \boldsymbol{u})$
16: $\quad \boldsymbol{x}^e \leftarrow$ `new environment state` ▷ Observation
17: $\quad t \leftarrow t + \Delta$
18: **end while**

VII. EXPERIMENTS

Algorithm 1 is planner agnostic, but we have chosen to implement it using Via-point-based Stochastic Trajectory Optimization (VP-STO) [26] to solve Problem 2, a state-of-the-art motion planning and RHC algorithm. VP-STO parametrizes trajectories using *via points* that the robot passes through. VP-STO uses Covariance Matrix Adaptation (CMA-ES) [27] as the black box optimization technique. CMA-ES is a zero-order optimization method and thus can handle discontinuous objective functions, making it a good fit for RoSI. For CMA-ES parameters, we use a populations size of $\lambda = 10$ and an initial distribution $\mathcal{N}(\mathbf{0}_{2\times 1}, 10\boldsymbol{I}_{2x2})$. To calculate the RoSI, we use the tool `stlrom` [28].

As a baseline, we have implemented the MICP approach to shrinking and receding horizon control from [5]. We formulate and solve the MICP using the state-of-the-art approach from [13] including their library `stlpy`, which leverages Gurobi [29] and Drake [30] for efficient optimization.

For our robot dynamics, we consider a two-dimensional double integrator, i.e., $\boldsymbol{x}_t^r = [p_x, p_y, v_x, v_y]^T$, $\boldsymbol{u}_t = [a_x, a_y]^T$, and

$$\boldsymbol{x}_{t+\Delta}^r = \tilde{f}(\boldsymbol{x}_t^r, \boldsymbol{u}_t) = \begin{bmatrix} \boldsymbol{I}_{2x2} & \Delta\boldsymbol{I}_{2x2} \\ \mathbf{0}_{2x2} & \boldsymbol{I}_{2x2} \end{bmatrix} \boldsymbol{x}_t^r + \begin{bmatrix} 0.5\Delta^2\boldsymbol{I}_{2x2} \\ \boldsymbol{I}_{2x2} \end{bmatrix} \boldsymbol{u}_t,$$

where Δ is the sampling time. The environment for all case studies is that shown in Fig. 1, where the circle with radius 0.5 represents the dynamic environment. The predicate for describing when the robot is inside the dynamic circle is given as `region` in Fig. 1 for the NLP approach. The MICP approach cannot handle nonlinear predicates, and so `region` is instead approximated by a square inscribed within the circle, described as a conjunction of linear predicates. The robot is subject to velocity constraints of 2 m/s and input constraints of 2 m/s^2 in both dimensions. For each experiment, the robot begins at $\mathbf{x}_0^r = [4.5, 4.5]^T$.

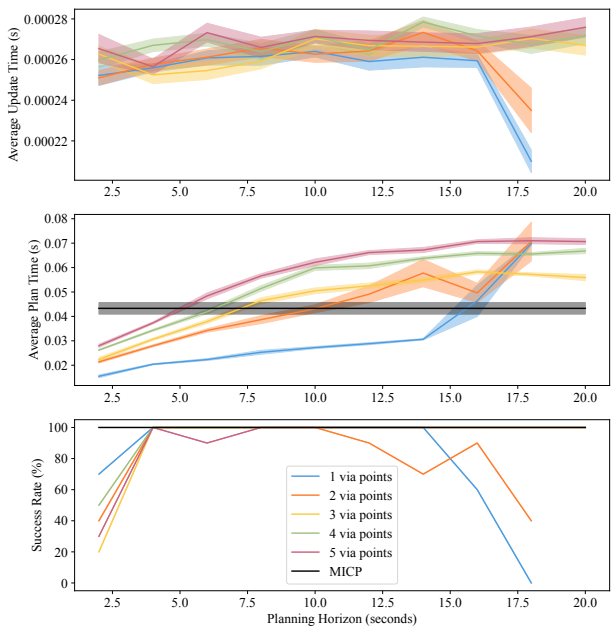


Fig. 2: Impact of plan horizon on controller performance for φ_{stayIn}.

To evaluate our approach against real-time requirements, we have implemented the algorithm in a simulated environment in the Robot Operating System (ROS) [31]. The simulation runs at 50 Hz, using $\Delta = 0.02$ for dynamics $\tilde{f}$. The dynamic circle $\boldsymbol{x}^e \in \mathbb{R}^2$ receives a random perturbation $\boldsymbol{w} = [w_1, w_2]^T$, where $w_1, w_2 \sim \mathcal{N}(0, 1)$, at this 50 Hz frequency. For each experiment, the environment begins at $\mathbf{x}_0^e = [2, 3]^T$. The planner runs concurrently to the application of control, and so the optimization problems must be solved within the allotted planning time for the solution to be applicable. Our NLP approach can plan reliably at 5 Hz ($\Delta = 0.2$), while the MICP approach plans at 1 Hz ($\Delta = 1$) due to the complexity of MICP. At each planning occurrence, the robot assumes no knowledge of the environmental disturbances and plans for a static world. The response to the dynamic environment comes from the closed-loop feedback of the RHC approach.

A. Case Study 1: Impacts of Planning Horizon for a Bounded Specification

The first experiment investigates the impact of the planning horizon h on the performance of Algorithm 1. We examine the (bounded) task described by φ_{stayIn} from Fig. 1, and test a range of horizons for calculating the RoSI. We are interested in the impacts of the planning horizon on the controller success rate, average plan time, and the time it takes to update the RCPE formula. For each planning horizon, we also test a range of via points used in VP-STO. As the baseline, we implement the MICP shrinking horizon controller. Fig. 2 presents the results of this experiment. For each planning horizon and via point count combination, we report the mean and standard error of the mean over ten simulations.

Specification	RHC Approach	Average Update Time (s)	Average Plan Time (s)	Success Rate (%)
$\varphi_{delivery}$	Algorithm 1	6.2E-4 $\pm$ 5.5E-5	0.062 $\pm$ 0.006	98
	MICP	–	0.234 $\pm$ 0.024	92
$\varphi'_{delivery}$	Algorithm 1	7.6E-4 $\pm$ 1.1E-4	0.081 $\pm$ 0.017	60
	MICP	–	–	–

TABLE I: Alg. 1 performance for unbounded specifications.

We see that, regardless of the number of via points, planning time generally increases with planning horizon. The performance of our approach with longer planning horizons is comparable to the baseline unless it has too few via points. VP-STO generates trajectories that pass through via points using splines, and the planner cannot find satisfying motion plans at long horizons when considering plans parametrized by only one or two via points. The performance of our approach with short planning horizons varies. This experiment reveals that, if the planning horizon is shorter than the memory of the formula, our approach lacks the foresight to understand how each point in memory impacts the robustness. This hurts the performance of the controller, with regards to success rate. For this reason, we recommend planning with $h \geq m(\varphi)$. With planning horizons $m(\varphi) < h < [\varphi]^{\uparrow}$, our approach performs comparably to the baseline with regards to success rate, and can even manage to outperform it with regards to average plan time. We also note that the average RCPE update time is on the order of 1% of the average plan time, and is not a bottleneck in Algorithm 1. Fig. 1 shows three planning instances from one RHC run using one via point and $h = 5$.

B. Case Study 2: Receding Horizon Control for Unbounded Specifications

The second experiment investigates the performance of our approach for unbounded specifications. We examine the tasks

$$\begin{aligned}\varphi_{delivery} =& \square_{[0,T_{max})}\big[(\texttt{goal} \rightarrow \lozenge_{[0,10]}\texttt{region}) \\ & \wedge (\texttt{region} \rightarrow \lozenge_{[0,10]}\texttt{goal})\big], \\ \varphi'_{delivery} =& \varphi_{delivery} \quad \wedge \lozenge_{[20,30]}(\mathbf{x}^r[0] > 4 \wedge \mathbf{x}^r[1] < 1),\end{aligned}$$

where formulas `goal` and `region` are as defined and shown in Fig. 1. The formula $\varphi_{delivery}$ describes the application from the Section I, where the robot is moving between the truck and human every ten seconds. The formula $\varphi_{delivery}$ is a safety specification, i.e., $\varphi_{delivery} \in \Phi_{\text{Uni-Temporal}}$, allowing for the MICP RHC approach presented in [5]. In contrast, $\varphi'_{delivery} \notin \Phi_{\text{Uni-Temporal}}$ is a more general unbounded specification for which that approach does not work. We note that these formulas are unbounded if $T_{max} = \infty$, but for benchmarking purposes we use $T_{max} = 80$. For our approach, we use planning horizon $h = m(\varphi_{delivery}) = m(\varphi'_{delivery}) = 10$ seconds and two via points for VP-STO.

Table I reports the mean and standard error of the mean over fifty simulations of each approach and formula combination. For $\varphi_{delivery}$, our approach performs comparably to the baseline, and outperforms it slightly in average plan time and success rate. Our approach also finds moderate success on $\varphi'_{delivery}$, for which the baseline approach cannot plan.

Across both case studies, we note that our approach performs comparably to the baseline with regards to success rate while outperforming it in planning time. We also note that our approach is more general than the baseline, considering a larger set of unbounded formulas than the baseline while allowing nonlinear systems and predicates.

VIII. CONCLUSIONS

In this work, we propose bounded-memory receding horizon control for STL specifications. We use techniques from runtime verification to constrain planning to a finite horizon. We extend syntactic separation of MTL formulas to STL to identify a subset of STL for which we can monitor the STL robustness using bounded memory by partially evaluating formulas. Our approach extends the subset of STL for which receding horizon control can be applied. For future work, we intend to apply this partial evaluation approach to other monitoring techniques, such as formula progression [32].

REFERENCES

[1] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[2] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *International symposium on formal techniques in real-time and fault-tolerant systems*. Springer, 2004, pp. 152–166.

[3] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2010, pp. 92–106.

[4] R. D. McAllister and J. B. Rawlings, "The stochastic robustness of nominal and stochastic model predictive control," *IEEE Transactions on Automatic Control*, vol. 68, no. 10, pp. 5810–5822, 2022.

[5] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 81–87.

[6] Y. Gilpin, V. Kurtz, and H. Lin, "A smooth robustness measure of signal temporal logic for symbolic control," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 241–246, 2020.

[7] J. V. Deshmukh, A. Donzé, S. Ghosh, X. Jin, G. Juniwal, and S. A. Seshia, "Robust online monitoring of signal temporal logic," *Formal Methods in System Design*, vol. 51, pp. 5–30, 2017.

[8] P. Hunter, J. Ouaknine, and J. Worrell, "Expressive completeness for metric temporal logic," in *2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*. IEEE, 2013, pp. 349–357.

[9] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimization-based trajectory generation with linear temporal logic specifications," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 5319–5325.

[10] C. Belta and S. Sadraddini, "Formal methods for control synthesis: An optimization perspective," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, no. 1, pp. 115–140, 2019.

[11] S. S. Farahani, V. Raman, and R. M. Murray, "Robust model predictive control for signal temporal logic synthesis," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 323–328, 2015.

[12] S. Sadraddini and C. Belta, "Robust temporal logic model predictive control," in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2015, pp. 772–779.

[13] V. Kurtz and H. Lin, "Mixed-integer programming for signal temporal logic with fewer binary variables," *IEEE Control Systems Letters*, 2022.

[14] H. Abbas and G. Fainekos, "Computing descent direction of mtl robustness for non-linear systems," in *2013 American Control Conference*. IEEE, 2013, pp. 4405–4410.

[15] Y. V. Pant, H. Abbas, and R. Mangharam, "Smooth operator: Control using the smooth robustness of temporal logic," in *2017 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2017, pp. 1235–1240.

[16] Y. V. Pant, H. Abbas, R. A. Quaye, and R. Mangharam, "Fly-by-logic: Control of multi-drone fleets with temporal logic objectives," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2018, pp. 186–197.

[17] R. Takano, H. Oyama, and M. Yamakita, "Continuous optimization-based task and motion planning with signal temporal logic specifications for sequential manipulation," in *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2021, pp. 8409–8415.

[18] N. Mehdipour, C.-I. Vasile, and C. Belta, "Arithmetic-geometric mean robustness for control from signal temporal logic specifications," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 1690–1695.

[19] L. Lindemann and D. V. Dimarogonas, "Robust control for signal temporal logic specifications using discrete average space robustness," *Automatica*, vol. 101, pp. 377–387, 2019.

[20] E. A. Gol, "Efficient online monitoring and formula synthesis with past stl," in *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE, 2018, pp. 916–921.

[21] C.-I. 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)*. IEEE, 2017, pp. 3840–3847.

[22] X. Yu, C. Wang, D. Yuan, S. Li, and X. Yin, "Model predictive control for signal temporal logic specifications with time interval decomposition," in *2023 62nd IEEE Conference on Decision and Control (CDC)*. IEEE, 2023, pp. 7849–7855.

[23] Z. Zhang and S. Haesaert, "Modularized control synthesis for complex signal temporal logic specifications," in *2023 62nd IEEE Conference on Decision and Control (CDC)*. IEEE, 2023, pp. 7856–7861.

[24] P. Bouyer, U. Fahrenberg, K. G. Larsen, N. Markey, J. Ouaknine, and J. Worrell, "Model checking real-time systems," *Handbook of model checking*, pp. 1001–1046, 2018.

[25] H. Abbas, Y. V. Pant, and R. Mangharam, "Temporal logic robustness for general signal classes," in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, 2019, pp. 45–56.

[26] J. Jankowski, L. Brudermüller, N. Hawes, and S. Calinon, "Vp-sto: Via-point-based stochastic trajectory optimization for reactive robot behavior," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 10 125–10 131.

[27] N. Hansen, "The cma evolution strategy: A tutorial."

[28] A. Donzé, Decyphir et. al. Stlrom. [Online]. Available: https://github.com/decyphir/STLRom

[29] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2024. [Online]. Available: https://www.gurobi.com

[30] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," 2019. [Online]. Available: https://drake.mit.edu

[31] Stanford Artificial Intelligence Laboratory et al., "Robotic operating system." [Online]. Available: https://www.ros.org

[32] F. Bacchus and F. Kabanza, "Planning for temporally extended goals," *Annals of Mathematics and Artificial Intelligence*, vol. 22, pp. 5–27, 1998.