Proceedings of Machine Learning Research vol vvv:1–14, 2027

Skip the Solve: Learning-Accelerated Recursively Feasible Interval-STL for Runtime Assurance

Evanns Morales-Cuadrado[*,†,1] EGM@GATECH.EDU
Xinpei Ni[*,1] XNI32@GATECH.EDU
Luke Baird[3] LUKE.BAIRD@RTX.COM
Glen Chou[2] CHOU@GATECH.EDU
Samuel Coogan[1] SAM.COOGAN@GATECH.EDU
Georgia Institute of Technology: [1]School of Electrical and Computer Engineering, [2]Schools of Cybersecurity & Privacy and Aerospace Engineering, [3]Raytheon

Abstract

Autonomous robots must often balance mission performance with temporal specifications involving deadlines, restricted regions, and recurring obligations despite uncertainty and limited computational resources. For long-term missions subject to disturbances, receding-horizon planning must ensure that each trajectory admits a continuation that preserves specification satisfaction at future planning instances. Interval signal temporal logic (I-STL) can capture uncertainty in system behavior and specification predicates, while persistently feasible terminal constraints can ensure recursive feasibility. However, existing synthesis methods repeatedly solve costly mixed-integer programs, limiting their use on resource-constrained robots. We propose Learning-augmented Interval-STL for Recursively-feasible Runtime Assurance (LIRRA), which accelerates I-STL synthesis by verifying learned trajectory proposals against the original synthesis constraints, including control-invariant terminal conditions. If a proposal is rejected, LIRRA attempts delay-compensated fallback synthesis and follows the last verified trajectory, which provides a specification-satisfying continuation even if synthesis fails. Together, these mechanisms ensure recursive feasibility and specification satisfaction under bounded dynamics and predicate uncertainty, independently of the learned policy and solver completion time. We evaluate LIRRA on a MuJoCo Menagerie Skydio X2 quadrotor and a custom MuJoCo blimp model, demonstrating substantial reductions in synthesis frequency, CPU energy consumption, and computational latency.

Keywords: Signal temporal logic, runtime assurance, recursive feasibility, amortized optimization, learning-based control

1. Introduction

Autonomous robotic systems must often satisfy safety specifications involving deadlines, time windows, and recurring obligations that are highly cumbersome to express through conventional state and input constraints. Signal temporal logic (STL) provides a formal framework for expressing such requirements through temporal operators (Maler and Nickovic, 2004). In this work, we use STL to express safety requirements that must be enforced at every step in a receding-horizon manner, executing an initial portion of each plan before replanning from an updated state. A key challenge of the receding-horizon setting is that every specification-enforcing trajectory segment must also preserve recursive feasibility. That is, satisfying the specification over the current planning horizon is insufficient if the trajectory leads the system to a configuration from which future violations are unavoidable. Feasible continuations must therefore remain available after every planning step.

∗. Equal contribution.
†. Corresponding author.

© 2027 E. Morales-Cuadrado, X. Ni, L. Baird, G. Chou & S. Coogan.

Uncertainty introduces an additional challenge, as nominal satisfaction does not guarantee satisfaction under disturbances or modeling errors. Uncertainty may also arise in the specification itself, such as when the boundaries of a restricted region are only known within prescribed bounds. Interval signal temporal logic (I-STL) addresses both forms of uncertainty through interval-valued signals and predicates, allowing satisfaction to be verified over sets of possible trajectories and specification realizations (Baird et al., 2023). More recently, Baird et al. (2026) introduced a receding-horizon I-STL runtime assurance framework that combines reasoning over bounded uncertainty with persistently feasible terminal conditions to ensure recursive feasibility and sustained specification satisfaction. However, despite these guarantees, synthesis-based I-STL runtime assurance requires solving a mixed-integer linear program (MILP) (Raman et al., 2014; Kurtz and Lin, 2022) at every planning step, imposing a considerable computational burden on resource-constrained robotic platforms. Indeed, due to the weight limitations on the autonomous blimp platform of (Baird et al., 2026), no computing platform powerful enough to adequately solve such programs was able to fit onboard, and computation was forced to shift to an offboard ground control station. In contrast, learned controllers can transfer much of this burden to offline training. Neural networks can approximate model predictive controllers, effectively amortizing repeated optimization (Chen et al., 2018; Hertneck et al., 2018; Cauligi et al., 2022), while reinforcement and self-supervised learning can train policies to satisfy STL specifications or maximize their robustness (Aksaray et al., 2016; Venkataraman et al., 2020; Meng and Fan, 2023). However, approximation errors may compromise specification satisfaction, motivating architectures whose formal guarantees do not depend on learning accuracy.

These observations motivate an architecture that combines the runtime efficiency of learning-based control with the formal guarantees of the receding-horizon I-STL planning framework. In particular, we adopt a runtime assurance (RTA) approach that seeks to preserve the behavior of a mission-driven nominal controller while ensuring specification satisfaction. To this end, we propose Learning-augmented Interval-STL for Recursively-feasible Runtime Assurance (LIRRA), which reduces online optimization through learned proposals while preserving recursive feasibility and specification satisfaction under bounded dynamical and predicate uncertainty.

Related work. Robust and stochastic STL model predictive control accounts for bounded disturbances through predicate tightening (Sadraddini and Belta, 2015; Ma et al., 2026), but recursive feasibility is not generally guaranteed. Sadraddini and Belta (2019) establish recursive feasibility for positive monotone systems using robust control-invariant terminal sets and repeated MILP synthesis. Triggered STL MPC (Huang et al., 2024; Lin and Baras, 2020) and learning-based STL control (Meng and Fan, 2023; Liu et al., 2023) reduce online computation but do not guarantee recursive feasibility.

More closely related are predictive safety filters, which certify learned control inputs through online robust MPC optimization under bounded disturbances (Wabersich and Zeilinger, 2018), and Hose et al. (2025), who avoid online MPC optimization by verifying learned input sequences. Model predictive shielding (Bastani, 2021), gatekeeper architectures (Agrawal et al., 2024), and reachability-based trajectory design (Kousik et al., 2020; Morales-Cuadrado et al., 2026) similarly verify decisions against backup or fail-safe behavior. However, none of these approaches combines learning-accelerated runtime assurance with recursively feasible satisfaction of temporal specifications under bounded uncertainty in both dynamics and predicates.

Contributions. The main contributions are threefold:[1] (i) a learning-accelerated I-STL runtime assurance architecture that utilizes verified trajectory proposals, cached feasible continuations, and delay-compensated fallback to preserve recursive feasibility and specification satisfaction under bounded dynamical and predicate uncertainty independently of learning accuracy and solver completion time; (ii) a training framework combining behavior cloning, dataset aggregation (DAgger), and PPO under bounded uncertainty; and (iii) an experimental evaluation of specification satisfaction, synthesis frequency, computational latency, CPU energy consumption, and mission performance on three-dimensional, uncertainty-aware extensions of the patrol specifications of Baird et al. (2026) on a MuJoCo Skydio X2 quadrotor and a new custom MuJoCo model of Georgia Tech's Miniature Autonomous Blimp platform (GT-MAB) (Tao et al., 2020). Indeed, to the best of our knowledge, LIRRA is the first framework to use learning to reduce online receding-horizon STL synthesis while ensuring recursive feasibility and specification satisfaction under set-bounded uncertainty in both dynamics and predicates.

2. Preliminaries

We introduce the planning model and the interval STL specifications used throughout.

System model and uncertainty. We use the discrete-time planning model

$$x_{k+1} = Ax_k + Bu_k + Gw_k, \qquad x_k \in \mathcal{X}, \quad u_k \in \mathcal{U}, \quad w_k \in \mathcal{W}, \tag{1}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{U} \subseteq \mathbb{R}^m$ are state and input constraint boxes, respectively. The full state x_k is measured, and $y_k = Cx_k \in \mathbb{R}^p$ denotes the output on which specifications are evaluated. We denote closed and bounded boxes in $\mathbb{R}^s$ by $[a, b] \coloneqq \{z \in \mathbb{R}^s : a \preceq z \preceq b\}$, where $\preceq$ denotes the componentwise vector order, so that $x \preceq y$ if and only if $x_i \leq y_i$ for all $i \in \{1, \ldots, s\}$. The disturbance $w_k \in \mathcal{W} = [-\bar{w}, \bar{w}] \subseteq \mathbb{R}^q$ enters through $G \in \mathbb{R}^{n\times q}$ and can capture external disturbances as well as modeling errors.

A nominal plan $(\hat{x}_{0:N}, \hat{u}_{0:N-1})$ satisfies $\hat{x}_{i+1} = A\hat{x}_i + B\hat{u}_i$, where i indexes steps relative to the planning root. At absolute step k, let $\hat{x}_k$ and $\hat{u}_k$ denote the corresponding state and input of the active plan. We track this plan using $u_k = \hat{u}_k + K(x_k - \hat{x}_k)$, with K chosen so that $A_K = A + BK$ is Schur stable. The tracking error $e_k = x_k - \hat{x}_k$ then evolves as $e_{k+1} = A_K e_k + Gw_k$. Initializing the nominal trajectory at a measured state sets the tracking error to zero. We call the number of steps elapsed since this initialization the *error age* ℓ. At age ℓ, the error lies in the disturbance-reachable set $\mathcal{Z}(\ell) = \bigoplus_{j=0}^{\ell-1} A_K^j G\mathcal{W}$, where $\bigoplus$ denotes repeated Minkowski summation. The interval hull of $\mathcal{Z}(\ell)$ is $[-\bar{z}(\ell), \bar{z}(\ell)]$, where $\bar{z}(\ell) = \sum_{j=0}^{\ell-1} |A_K^j G|\bar{w}$, and $|\cdot|$ denotes the elementwise absolute value. Similarly, the feedback correction lies in $[-\bar{k}(\ell), \bar{k}(\ell)]$, where $\bar{k}(\ell) = \sum_{j=0}^{\ell-1} |KA_K^j G|\bar{w}$ (Chisci et al., 2001; Mayne et al., 2005). We use these bounds to tighten the state and input constraints, so that verifying the nominal plan ensures constraint satisfaction for all disturbances in $\mathcal{W}$.

Temporal specifications under uncertainty. We express temporal requirements using STL formulas ψ constructed from affine predicates $\mu : a^\top y - b \geq 0$ and bounded temporal operators. The robustness $\rho_\psi(\mathbf{y}, k)$ of an output signal $\mathbf{y}$ quantifies satisfaction through signed predicate margins combined using minima and maxima over logical and temporal operators (Donzé and Maler, 2010).

1. Project website: https://evannsmc.github.io/rl-stl/.

I-STL[2] extends robustness evaluation to interval-valued signals and predicates, accounting for uncertainty in both system trajectories and specification parameters. We therefore allow predicates $[a]^\top y - [b] \geq 0$, where $[a]$ and $[b]$ are intervals of bounded uncertainty. For a predicted state $\hat{x}_k$ with error age ℓ, let $[y_k] = [C\hat{x}_k - |C|\bar{z}(\ell),\ C\hat{x}_k + |C|\bar{z}(\ell)]$. Robust satisfaction of the predicate with margin $\delta \geq 0$ requires $\underline{[a]^\top [y_k] - [b]} \geq \delta$, where the underline denotes the lower interval endpoint. For exact predicates, this reduces to $a^\top C\hat{x}_k - b \geq \delta + |a^\top C|\bar{z}(\ell)$.

3. Methodology

Let ψ have horizon h, so that its satisfaction at step s depends only on the *window* of $h+1$ consecutive outputs $y_{s:s+h}$.

Problem statement. Given a proposer π generating candidate input sequences $\tilde{u}_{0:N-1}$, we seek a runtime assurance architecture that (i) executes only verified control sequences to guarantee satisfaction of ψ on every window $y_{s:s+h}$, $s \geq 0$, under disturbances $w_k \in \mathcal{W}$, and predicate parameters over $[a]$ and $[b]$; (ii) preserves feasible continuations despite rejected proposals or failed synthesis; and (iii) reduces online optimization by accepting feasible proposals. Requirements (i)–(ii) must hold independently of the proposer and solver completion time, under prescribed assumptions.

3.1. Proposed Method: LIRRA

We propose Learning-augmented Interval-STL for Recursively-feasible Runtime Assurance (LIRRA), which replaces repeated online synthesis with verified learned trajectory proposals.

3.1.1. RECEDING-HORIZON SYNTHESIS

We extend the I-STL runtime assurance formulation of Baird et al. (2026) by tracking error ages to support plans rooted at predicted future states for delay-compensated synthesis. For a planning root at absolute step t, the synthesis problem considers the preceding $h-1$ outputs, the nominal root $\hat{x}_0$, and an N-step nominal trajectory $\hat{x}_{1:N}$. Here, i indexes steps relative to the root, so $\hat{x}_i$ corresponds to absolute step $t+i$. The history is necessary because windows starting before t contain realized samples that constrain the plan; i.e., a region entered before t may have to be exited by a deadline after t. Each planned sample $\hat{x}_i$, $i = 1, \ldots, N$, ends the window $[t+i-h, t+i]$. Accordingly, the *enforced windows*, on which $\mathcal{P}$ requires ψ to hold, start at $t-h+1, \ldots, t-h+N$. Windows beyond the horizon are handled by terminal conditions.

The nominal root $\hat{x}_0$ is initialized at either the measured state x_t with error age $o = 0$ or a nominal state from the cached plan with accumulated error age o. We refer to the most recently verified plan, which is stored and tracked until a verified replacement is available, as the *cached plan*. Its nominal states provide *cached predictions* of future system states. A prediction i steps beyond the root has error age $\ell = o + i$. Because MILP synthesis takes time, we allocate a computation budget of d steps and root the problem at the cached prediction $\hat{x}_{k+d}$, so that its solution is ready by the step it is applied. The history preceding the future root combines available measurements with cached predictions of unobserved samples, which retain their original error ages to preserve the output interval bounds. Because measured values are fixed while predictions may deviate from the realized trajectory, we define ρ^{I}_δ by subtracting δ from the lower interval predicate robustness

2. For a complete treatment of I-STL and its syntax, grammar, and semantics, please see (Baird et al., 2023).

of predicted samples only, including those in the history. Thus, $\rho^{\mathrm{I}}_{\delta} \geq 0$ requires the margin δ on predictions and nonnegative robustness on measurements, ensuring robust satisfaction on predicted states while avoiding conservative margins on already-safe observed states.

Given the nominal root $\hat{x}_0$ with error age o, preceding history $\mathcal{H} = (y_{t-h+1}, \ldots, y_{t-1})$, and a nominal mission trajectory $(x^{\mathrm{nom}}, u^{\mathrm{nom}})$, we formulate the receding-horizon synthesis problem

$$
\begin{aligned}
\mathcal{P}(\hat{x}_0, o, \mathcal{H}): \quad \min_{\hat{u}_{0:N-1}} \; & \textstyle\sum_{i=1}^{N_c} \gamma^i \|W(\hat{x}_i - x^{\mathrm{nom}}_i)\|_1 + w_u \sum_{i=0}^{N-1} \|\hat{u}_i - u^{\mathrm{nom}}_i\|_1 \\
\text{s.t.} \quad & \hat{x}_{i+1} = A\hat{x}_i + B\hat{u}_i, \quad i = 0, \ldots, N-1, \\
& \hat{x}_i \in \mathcal{X} \ominus [-\bar{z}(o+i), \bar{z}(o+i)], \quad i = 0, \ldots, N, \\
& \hat{u}_i \in \mathcal{U} \ominus [-\bar{k}(o+i), \bar{k}(o+i)], \quad i = 0, \ldots, N-1, \\
& \rho^{\mathrm{I}}_{\delta}(\mathcal{H}, \hat{x}_{0:N}, s) \geq 0, \quad s = \max(0, t-h+1), \ldots, t-h+N, \\
& \hat{x}_{N-\beta+1}, \ldots, \hat{x}_N \in \mathcal{Q}_f \quad \text{for some } f,
\end{aligned}
\tag{2}
$$

where $\rho^{\mathrm{I}}_{\delta}(\mathcal{H}, \hat{x}_{0:N}, s)$ is the interval robustness of ψ on the window starting at each s, whose samples before the root are taken from $\mathcal{H}$ and the rest from the planned output intervals $[\hat{y}_{t+i}]$, $\mathcal{Q}_1, \ldots, \mathcal{Q}_F$ are the verified terminal sets of Section 3.1.2, and $\beta \geq 1$ is the number of final states that must lie in a common $\mathcal{Q}_f$. The objective weighs state and input deviations from the nominal mission by $W \succeq 0$ and $w_u \geq 0$, with discount factor $\gamma \in (0, 1]$. It may cover only the first $N_c \leq N$ planned states, while the full horizon N must accommodate the enforced windows and a terminal block that admits a specification-satisfying continuation (Baird et al., 2026). The terminal block guarantees such a continuation for every feasible plan, so the system is never stranded in a state from which violations are inevitable, and synthesis remains feasible at later planning steps. For linear dynamics and affine interval predicates with fixed uncertainty bounds, $\mathcal{P}$ is a MILP.

3.1.2. CONTROL-INVARIANT TERMINAL SETS

If a proposal is rejected, and fallback synthesis returns no verified plan within the available computation time, the previous cached plan must be continued beyond its horizon N. Problem $\mathcal{P}$ therefore ends each plan in a common terminal set $\mathcal{Q}_f$, $f \in \{1, \ldots, F\}$, where each $\mathcal{Q}_f$ and terminal controller κ_f satisfy (C1) $\mathcal{Q}_f \subseteq \mathcal{X} \ominus [-\bar{z}(\infty), \bar{z}(\infty)]$; (C2) for every $x \in \mathcal{Q}_f$, $Ax + B\kappa_f(x) \in \mathcal{Q}_f$ and $\kappa_f(x) \in \mathcal{U} \ominus [-\bar{k}(\infty), \bar{k}(\infty)]$; and (C3) for any feasible plan of $\mathcal{P}$ whose nominal trajectory remains in $\mathcal{Q}_f$ from step $N - \beta + 1$ onward, every $y_{s:s+h}$ window with $s + h > t + N$ satisfies its tightened predicate conditions.

Lemma 1 *Under (C1)–(C3), any feasible plan of $\mathcal{P}$ whose final β nominal states lie in a common $\mathcal{Q}_f$ admits the indefinite continuation $\hat{u}_i = \kappa_f(\hat{x}_i)$, $i \geq N$, which satisfies the tightened state and input constraints at every error age and every window of ψ.*

Proof By (C2), the continuation remains in $\mathcal{Q}_f$ with inputs satisfying the infinite-age input tightening, and by (C1) its states satisfy the infinite-age state tightening. Since $\bar{z}$ and $\bar{k}$ are nondecreasing, the infinite-age tightenings are at least as restrictive as those at any finite age. Windows ending by step N are certified by $\mathcal{P}$, and those extending beyond N by (C3). ■

3.1.3. RUNTIME VERIFICATION AND FALLBACK

The cached plan $\mathcal{C} = (\hat{x}, \hat{u}, r, o_r, f)$ stores nominal states and inputs, root step r, root error age o_r, and terminal-set index f. If the next proposal is accepted, $\mathcal{C}$ is replaced after one step; otherwise,

Algorithm 1: LIRRA: Verified proposals with delay-compensated I-STL fallback

Input: proposer π, program $\mathcal{P}$, budget d, gain K

1 $\mathcal{C} \leftarrow \text{solve}(\mathcal{P}(x_0, 0, \emptyset))$ at $t = 0^-$;
2 **for** $k = 0, 1, \ldots$ **do**
3 measure x_k and update $\mathcal{H}_k$;
4 **if** *a solve is due at* k **then**
5 $\mathcal{C} \leftarrow$ its plan if it meets the constraints of its $\mathcal{P}$ and returned on time;
6 **if** *no solve pending and* $\mathcal{C}$ *not replaced at* k **then**
7 $\tilde{u} \leftarrow \pi(x_k, \mathcal{H}_k)$, $\tilde{x} \leftarrow \text{rollout}(x_k, \tilde{u})$;
8 **if** $(\tilde{x}, \tilde{u})$ *feasible for* $\mathcal{P}(x_k, 0, \mathcal{H}_k)$ **then**
9 $\mathcal{C} \leftarrow (\tilde{x}, \tilde{u}, k, 0, f)$;
10 **else**
11 start $\mathcal{P}(\hat{x}_{k+d}, o_r + k + d - r, \mathcal{H}_k^d)$, due at $k + d$, warm-started from $\mathcal{C}(k + d)$;
12 $(\hat{x}_k, \hat{u}_k) \leftarrow \mathcal{C}(k)$ and apply $u_k = \hat{u}_k + K(x_k - \hat{x}_k)$;

as shown in Figure 1, cached execution continues for the d-step computation budget while fallback synthesis is pending. If fallback synthesis also fails, the remainder of the cached plan reaches a terminal set, where κ_f preserves safety indefinitely beyond the plan's horizon even if all subsequent MILP-based synthesis attempts fail. A new proposal replaces $\mathcal{C}$ only if it passes an *acceptance check*. That is, if its nominal rollout $\tilde{x}$ (Algorithm 1, line 7) is feasible for $\mathcal{P}(x_k, 0, \mathcal{H}_k)$, including its tube-tightened bounds. Because the inputs are fixed, this is a computationally lightweight constraint evaluation rather than a full MILP solve.

3.1.4. RECURSIVE FEASIBILITY AND SATISFACTION

The following theorem extends recursive-feasibility guarantees to verified trajectory replacements and delay-compensated fallback synthesis.

Theorem 1 *Let ψ be a bounded-horizon STL formula in negation normal form. Assume: (A1) the terminal sets and controllers satisfy (C1)–(C3); (A2) $\mathcal{P}(x_0, 0, \emptyset)$ is feasible; and (A3) the tracking error satisfies $|x_k - \hat{x}_k| \leq \bar{z}(o_r + k - r)$, where r and o_r are the cached plan's root step and error age. Then, for any proposer, Algorithm 1 (i) always supplies a feasible control input; (ii) requests only feasible fallback synthesis problems, admitting the remaining cached plan and its terminal extension as a candidate; and (iii) satisfies ψ on every window $y_s, \ldots, y_{s+h}$, $s \geq 0$, of the executed outputs, independently of solver completion time. For Equation (1) with $w_k \in \mathcal{W}$, the disturbance-tube construction implies (A3).*

Proof We prove by induction that the cached plan always admits a feasible infinite continuation. For (ii), suppose synthesis is requested at k from a cached plan rooted at r with error age o_r. To show that the fallback problem is feasible, we construct a candidate by following the cached trajectory from $k + d$ onward. This new root inherits age $o' = o_r + k + d - r$, and a sample at absolute step τ has error age $o' + \tau - (k + d) = o_r + \tau - r$, matching the age from the cache. Thus, the shifted cache retains the tightened constraints under which it was already verified. By Lemma 1, the terminal extension satisfies the tightened constraints and subsequent windows. In the preceding history, samples that remain predicted retain their certified margins, while (A3) places

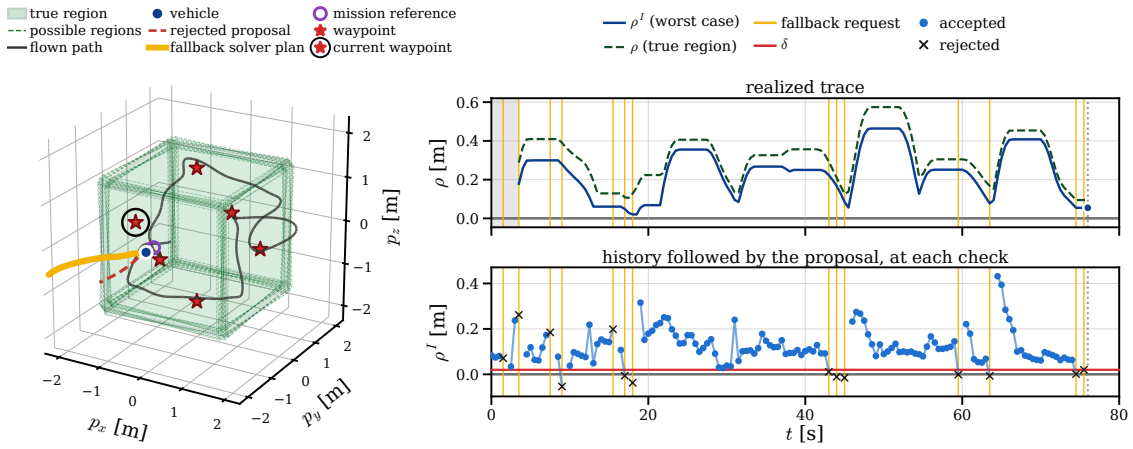


Figure 1: LIRRA on the GT-MAB under ϑ^+ with uncertain restricted region boundaries. After a rejected proposal (dashed red), the vehicle (blue dot) follows the newly synthesized and delay-compensated fallback plan (solid yellow).

newly measured outputs within their previously certified intervals. Replacing these predictions with measurements therefore preserves the predicate conditions supporting certified windows, since measurements require only nonnegative robustness. Consequently, the remaining cached trajectory and its terminal extension form a feasible candidate for the fallback problem, whether or not the solver returns a solution.

For (i), keeping the cached plan $\mathcal{C}$ or replacing it with a verified plan preserves this continuation property and supplies feasible inputs. For (iii), verification against the existing history preserves satisfaction of windows that began before the root, (A3) ensures that realized samples satisfy certified predicate conditions, and Lemma 1 covers windows beyond the planning horizon. For Equation (1) with $w_k \in \mathcal{W}$, (A3) follows from the disturbance-tube construction. ■

Remark 1 *Theorem 1 builds on the persistent-feasibility framework of Baird et al. (2026, Theorem 2). While that framework establishes recursive feasibility through persistently feasible terminal conditions, it does not explicitly account for tracking uncertainty accumulated during synthesis delays. Theorem 1 extends this guarantee to delayed fallback synthesis by initializing optimization at future predicted states and inheriting the accumulated tracking-error age, ensuring that the resulting trajectories remain feasible despite disturbances during the delay. Moreover, although our terminal-set construction is more restrictive than Baird et al. (2026), it allows us to provide an explicit continuation that preserves feasibility and specification satisfaction indefinitely.*

3.1.5. LEARNING-BASED TRAJECTORY GENERATION

We train multilayer perceptrons to propose input sequences over the fixed horizon N. Observations include the current state, preceding outputs and their predicate margins, the current waypoint, patrol task error, and the uncertainty bound.

Behavior cloning and DAgger. We initialize our proposer with labeled pairs $(\sigma, u^\star)$. Training states $\sigma = (x_t, y_{t-h+1:t-1})$ contain a system state and its preceding outputs. These are collected

from episodes driven by three controllers: the expert MILP controller, the expert with Gaussian input noise (Laskey et al., 2017), and the nominal mission controller, which ignores ψ. Each σ is labeled with the expert solution $u^\star = \hat{u}_{0:N-1}$ of $\mathcal{P}$ at σ, retaining only pairs that independently pass the acceptance check despite solver tolerances or early termination. We then apply DAgger (Ross et al., 2011), labeling states visited by the learned policy with expert solutions and retraining on the aggregated dataset. For a proposal $\tilde{u}$ with expert label $u^\star$, the cloning loss is

$$\mathcal{L}(\tilde{u}) = \frac{1}{Nm}\left\|\frac{\tilde{u}-u^\star}{\bar{u}}\right\|_2^2 + \lambda \max\big(0, \eta - c(\tilde{u})\big),$$

where m is the input dimension and $\bar{u}$ is the half-width of the tightened input box. The first term measures normalized imitation error; the second encourages a feasibility margin $c(\tilde{u})$ of at least η. This margin is a signed version of the acceptance check. That is, $c(\tilde{u}) \geq 0$ exactly when the proposal passes the check. For the rollout $\tilde{x}$ from the measured state, it is defined by

$$c(\tilde{u}) = \min\left(\rho_\delta^{\mathrm{I}},\ \max_f \min(g_f - H_f \tilde{x}_{N-\beta+1:N}),\ \min_{1\leq i\leq N} \min\big(\tilde{x}_i - x_{\min} - \bar{z}(i),\ x_{\max} - \bar{z}(i) - \tilde{x}_i\big)\right).$$

The first term is the interval robustness of Section 3.1.1, evaluated on the measured history followed by $\tilde{x}$ and minimized over all enforced windows. The second term is nonnegative exactly when the final β states lie in a common terminal set, from which κ_f continues the plan indefinitely (Lemma 1). For polytopic sets $\mathcal{Q}_f = \{x \mid H_f x \leq g_f\}$, it is the largest minimum slack over the final β states. The third term is the minimum slack of predicted states, including positions and velocities, to tube-tightened bounds of $\mathcal{X} = [x_{\min}, x_{\max}]$, so the realized state stays in $\mathcal{X}$ despite uncertainty.

Reinforcement learning. We fine-tune the cloned policy with PPO (Schulman et al., 2017) to reduce fallback frequency while maintaining mission performance, using Brax (Freeman et al., 2021) through the MuJoCo Playground (Zakka et al., 2025) interface. The reward at step k is

$$r = -c_J J - c_c \max(0, \eta - \max(c, -1)) + c_{\mathrm{ok}} \mathbf{1}[c \geq \eta] - c_s \mathbf{1}[c < 0] - c_v \mathbf{1}[\rho_\psi(y_{k-h+1:k+1}) < 0],$$

where J and c are evaluated at $\tilde{u}$, and J is the objective of Equation (2). These terms respectively penalize mission cost and insufficient feasibility margin, reward acceptance with margin, penalize rejection-induced synthesis, and penalize realized specification violations.

4. Experiments

We evaluate LIRRA on two MuJoCo vehicles: a MuJoCo Menagerie Skydio X2 quadrotor model (Zakka et al., 2022), and a newly developed MuJoCo model of Georgia Tech's Miniature Autonomous Blimp (GT-MAB) (Tao et al., 2020). [3]

Tasks and specifications. We build on the 2D patrol task of Baird et al. (2026) and extend it to 3D. In particular, both vehicles patrol around a restricted cube region $S = \{||p||_\infty < 1.41\}$ m, with positions measured from its center 2.75 m above the floor, visiting waypoints at ± 1.52 m along each axis in the order $+y, +x, +z, -y, -x, -z$, so each leg of the nominal patrol cuts through an edge region of S, motivating the specifications:

$$\vartheta = (x \in S) \Rightarrow \mathbf{F}_{[0,4]}(x \notin S), \qquad \varphi = (x \in S) \Rightarrow \mathbf{F}_{[0,3]}\mathbf{G}_{[0,1]}(x \notin S). \tag{3}$$

3. Code, videos, and supplementary material: https://evannsmc.github.io/rl-stl/.

Table 1: Comparison of LIRRA and standard MILP-based runtime assurance. Fallbacks denote synthesis requests per planning step; computation time and CPU energy are reported per planning step; waypoints measure mission progress; deviation is the mean distance from the nominal trajectory; violations count LIRRA episodes violating the specification.

System	Spec.	Fallbacks/step	Time [ms/step]		CPU energy [mJ/step]		Waypoints		Deviation [m]		Viol.
			MILP	LIRRA	MILP	LIRRA	MILP	LIRRA	MILP	LIRRA	
Skydio X2	ϑ	.006	140.9	**1.7**	1848	**13.7**	**6.0**	**6.0**	**0.09**	0.23	**0/30**
	φ	.007	151.5	**2.0**	2269	**18.2**	5.6	**6.0**	**0.21**	0.32	**0/30**
	ϑ^+	.013	158.8	**3.1**	2038	**51.7**	**6.0**	**6.0**	**0.12**	0.30	**0/30**
	φ^+	.011	151.2	**3.0**	2063	**38.5**	**6.0**	5.8	**0.20**	0.42	**0/30**
GT-MAB	ϑ	.062	10.9	**1.7**	148	**12.0**	3.0	**3.4**	0.72	**0.65**	**0/30**
	φ	.045	12.1	**1.8**	166	**11.9**	**3.2**	2.6	**0.65**	0.68	**0/29**
	ϑ^+	.077	11.7	**2.2**	146	**18.2**	2.0	**3.2**	0.82	**0.70**	**0/30**
	φ^+	.041	14.0	**1.9**	175	**15.8**	**2.0**	1.4	**0.78**	0.87	**0/23**

We also introduce specifications ϑ^+ and φ^+, which treat the boundaries of S as uncertain. The true region has a half-width in 1.41 ± 0.05 m and is rotated about its center by at most 0.05 rad about an unknown axis. Each face is thus defined by an interval predicate $[a]^\top y - [b] \geq 0$ with $[a]$ the interval hull of the rotated outward normal, so that I-STL satisfaction implies satisfaction for every admissible region. The 3D task is more challenging than its planar counterpart. S has six faces instead of four, the patrol includes vertical legs, and both its size and orientation are uncertain. Eight GT-MAB episodes yielded infeasible I-STL problems, violating (A2), and were excluded, leaving 29 episodes for φ and 23 for φ^+ in Table 1. No such cases occurred in the planar task.

We train MLP proposers with three 512-unit hidden layers. Each configuration is evaluated over 30 episodes per vehicle and formula, ten with each uncertainty level of $\varepsilon \in \{0.05, 0.10, 0.15\}$ m in the relevant dimensions, and a satisfaction margin of $\delta = 0.02$.

Terminal-set construction for the patrol tasks. For $S = \{p \in \mathbb{R}^3 : \|p\|_\infty < c\}$, we build one terminal set $\mathcal{Q}_f$ per face f of S. The planning dynamics decouple into two-state blocks $z_{a,j+1} = A_a z_{a,j} + B_a u_{a,j}$, one per axis a. For each f and a, we compute by predecessor iteration $\Omega \leftarrow \Omega \cap \mathrm{Pre}(\Omega)$ (Blanchini, 1999), under tightened input bounds, a control-invariant polygon $\Omega_{f,a}$ within the state box tightened by $\bar{z}(\infty)$ and, along the normal of f, beyond the face by a margin covering tracking error, predicate uncertainty, and δ. The product $\mathcal{Q}_f = \prod_a \Omega_{f,a}$ is then control invariant under the per-block terminal controller κ_f, and all its states lie outside S, so $\beta = 1$ for ϑ and $\beta = h_G + 1$ for φ satisfy (C3).

4.1. Results

Synthesis reduction and mission performance. Table 1 compares LIRRA with standard MILP-based runtime assurance across vehicles and specifications. For ϑ and φ, LIRRA required fallback synthesis on only 0.6–0.7% of X2 planning steps and 4.5–6.2% of GT-MAB steps. With uncertain boundaries, these were 1.1–1.3% and 4.1–7.7%, respectively. Although we report only the more challenging 3D tasks, these rates are comparable to those observed on the planar task, suggesting

that the proposer remains effective with increased spatial dimension and predicate uncertainty. Every LIRRA episode satisfied its temporal specification. On the quadrotor, LIRRA maintained the baseline pace of approximately six waypoints per episode. On the blimp, it reached more waypoints than baseline for ϑ and ϑ^+, but fewer for φ and φ^+. LIRRA deviated more from the nominal mission trajectory than the MILP baseline on the X2, and performed comparably on the blimp.

Although LIRRA prioritizes computational efficiency and specification satisfaction over mission performance, it can also improve patrol progress over the baseline, as on the blimp for ϑ and ϑ^+. A factor that may explain this advantage is that PPO maximizes mission reward over entire episodes, whereas each MILP solve minimizes tracking error over a finite horizon.

Ablation study. We perform an ablation study to assess the utility of our assurance mechanisms.

Proposal verification: without the acceptance check of line 8 of Algorithm 1, the proposer violated ϑ or φ in 57 of 238 episodes, including 48 of 60 blimp episodes for ϑ. Under the full acceptance check, the same proposer caused no violations, demonstrating that verification can preserve satisfaction despite proposer errors.

Nominal robustness trigger: we accept proposals whose nominal rollout robustness is at least δ, while ignoring tube tightening, state bounds, and the terminal condition. This variant caused no violations in 232 episodes and requested fewer GT-MAB fallback solves (460 versus 1541), but 90 returned no solution, compared with 3 under the full check. Each failure left the vehicle executing an unverified cached plan, removing guarantees of recursive feasibility and specification satisfaction.

Interval predicates: for ϑ^+ and φ^+, we evaluate each episode against a true restricted region drawn from the uncertainty set. The MILP baseline using nominal boundaries entered this region in 7 and 12 of 30 X2 episodes, respectively, and in 17 of 30 and 19 of 29 blimp episodes.

Delay compensation: initializing fallback synthesis at the measured state rather than the state predicted after the solve caused two episodes on each vehicle to violate the specification, and 31.4% of X2 and 26.2% of GT-MAB fallback solves returned no solution, compared with 0% and 0.2%, respectively with compensation enabled.

Computational cost and solver deadlines. LIRRA reduced mean computation time per planning step from 140.9–158.8 to 1.7–3.1 ms on the X2 and from 10.9–14.0 to 1.7–2.2 ms on the GT-MAB, yielding 50–83$\times$ and 5.3–7.4$\times$ speedups respectively. Fallback solves averaged 135–154 and 10.5–18.3 ms, respectively, versus 0.7–1.0 ms for inference and at most 0.6 ms for verification. The baseline missed seven X2 deadlines, while LIRRA missed none. LIRRA also reduced CPU energy per planning step from 1848–2269 to 13.7–51.7 mJ on the X2 and from 146–175 to 11.9–18.2 mJ on the GT-MAB, reductions of 39–135$\times$ and 8.0–13.9$\times$, respectively.

5. Conclusion

We propose a learning-accelerated I-STL runtime assurance architecture that reduces online synthesis through verified trajectory proposals while preserving recursive feasibility and specification satisfaction under bounded dynamical and predicate uncertainty. Evaluations on two robotic platforms with distinct dynamics demonstrate substantial reductions in computational latency and CPU energy consumption relative to conventional approaches with competitive mission performance and specification enforcement at all times.

Acknowledgments

This work was supported in part by the National Science Foundation under awards #2219755 and #2333488, and a Mathworks Microgrant.

References

Devansh R. Agrawal, Ruichang Chen, and Dimitra Panagou. gatekeeper: Online safety verification and control for nonlinear systems in dynamic environments. *IEEE Transactions on Robotics*, 40: 4358–4375, 2024. doi: 10.1109/TRO.2024.3454415.

Derya Aksaray, Austin Jones, Zhaodan Kong, Mac Schwager, and Calin Belta. Q-learning for robust satisfaction of signal temporal logic specifications. In *55th IEEE Conference on Decision and Control (CDC)*, pages 6565–6570, 2016. doi: 10.1109/CDC.2016.7799279.

Luke Baird, Akash Harapanahalli, and Samuel Coogan. Interval signal temporal logic from natural inclusion functions. *IEEE Control Systems Letters*, 7:3555–3560, 2023. doi: 10.1109/LCSYS.2023.3337744.

Luke Baird, Evanns Morales-Cuadrado, and Samuel Coogan. Runtime assurance for uncertain linear systems from interval signal temporal logic. SSRN preprint 6168587, 2026. URL https://papers.ssrn.com/sol3/papers.cfm?abstract_id=6168587.

Osbert Bastani. Safe reinforcement learning with nonlinear dynamics via model predictive shielding. In *American Control Conference (ACC)*, pages 3488–3494, 2021. doi: 10.23919/ACC50511.2021.9483182.

Franco Blanchini. Set invariance in control. *Automatica*, 35(11):1747–1767, 1999.

Abhishek Cauligi, Preston Culbertson, Edward Schmerling, Mac Schwager, Bartolomeo Stellato, and Marco Pavone. CoCo: Online mixed-integer control via supervised learning. *IEEE Robotics and Automation Letters*, 7(2):1447–1454, 2022. doi: 10.1109/LRA.2021.3135931.

Steven Chen, Kelsey Saulnier, Nikolay Atanasov, Daniel D. Lee, Vijay Kumar, George J. Pappas, and Manfred Morari. Approximating explicit model predictive control using constrained neural networks. In *American Control Conference (ACC)*, pages 1520–1527, 2018.

Luigi Chisci, J. Anthony Rossiter, and Giovanni Zappa. Systems with persistent disturbances: predictive control with restricted constraints. *Automatica*, 37(7):1019–1028, 2001.

Alexandre Donzé and Oded Maler. Robust satisfaction of temporal logic over real-valued signals. In *Formal Modeling and Analysis of Timed Systems*, volume 6246 of *Lecture Notes in Computer Science*, pages 92–106. Springer, 2010. doi: 10.1007/978-3-642-15297-9_9.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax – a differentiable physics engine for large scale rigid body simulation. In *Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

Michael Hertneck, Johannes Köhler, Sebastian Trimpe, and Frank Allgöwer. Learning an approximate model predictive controller with guarantees. *IEEE Control Systems Letters*, 2(3):543–548, 2018.

Henrik Hose, Johannes Köhler, Melanie N. Zeilinger, and Sebastian Trimpe. Approximate nonlinear model predictive control with safety-augmented neural networks. *IEEE Transactions on Control Systems Technology*, 33(6):2490–2497, 2025. doi: 10.1109/TCST.2025.3590268.

J. Huang, C. Wang, X. Yu, and X. Yin. Self-triggered model predictive control for signal temporal logic tasks. In *IEEE International Conference on Automation Science and Engineering (CASE)*, pages 2547–2552, 2024. doi: 10.1109/CASE59546.2024.10711435.

Shreyas Kousik, Sean Vaskov, Fan Bu, Matthew Johnson-Roberson, and Ram Vasudevan. Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots. *The International Journal of Robotics Research*, 39(12):1419–1469, 2020. doi: 10.1177/0278364920943266.

Vince Kurtz and Hai Lin. Mixed-integer programming for signal temporal logic with fewer binary variables. *IEEE Control Systems Letters*, 6:2635–2640, 2022. doi: 10.1109/LCSYS.2022.3172857.

Michael Laskey, Jonathan Lee, Roy Fox, Anca Dragan, and Ken Goldberg. DART: Noise injection for robust imitation learning. In *Conference on Robot Learning (CoRL)*, volume 78 of *Proceedings of Machine Learning Research*, pages 143–156, 2017.

Z. Lin and John S. Baras. Optimization-based motion planning and runtime monitoring for robotic agent with space and time tolerances. *IFAC-PapersOnLine*, 53(2):1874–1879, 2020. doi: 10.1016/j.ifacol.2020.12.2606.

Wenliang Liu, Wei Xiao, and Calin Belta. Learning robust and correct controllers from signal temporal logic specifications using BarrierNet. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 7049–7054. IEEE, 2023. doi: 10.1109/CDC49753.2023.10383857.

Liqian Ma, Zishun Liu, Glen Chou, and Yongxin Chen. Feedback motion planning for stochastic nonlinear systems with signal temporal logic specifications. *arXiv preprint arXiv:2605.02361*, 2026.

Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, volume 3253 of *Lecture Notes in Computer Science*, pages 152–166. Springer, 2004. doi: 10.1007/978-3-540-30206-3_12.

David Q. Mayne, María M. Seron, and Saša V. Raković. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41(2):219–224, 2005.

Yue Meng and Chuchu Fan. Signal temporal logic neural predictive control. *IEEE Robotics and Automation Letters*, 8(11):7719–7726, 2023. doi: 10.1109/LRA.2023.3315536.

Evanns Morales-Cuadrado, Long Kiu Chung, Shreyas Kousik, and Samuel Coogan. RTD-RAX: Fast, safe trajectory planning for systems under unknown disturbances. *arXiv preprint arXiv:2603.21635*, 2026.

Vasumathi Raman, Alexandre Donzé, Mehdi Maasoumy, Richard M. Murray, Alberto Sangiovanni-Vincentelli, and Sanjit A. Seshia. Model predictive control with signal temporal logic specifications. In *53rd IEEE Conference on Decision and Control (CDC)*, pages 81–87, 2014. doi: 10.1109/CDC.2014.7039363.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 627–635, 2011.

Sadra Sadraddini and Calin Belta. Robust temporal logic model predictive control. In *Annual Allerton Conference on Communication, Control, and Computing*, pages 772–779, 2015.

Sadra Sadraddini and Calin Belta. Formal synthesis of control strategies for positive monotone systems. *IEEE Transactions on Automatic Control*, 64(2):480–495, 2019. doi: 10.1109/TAC.2018.2814631.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Qiuyang Tao, Mengxue Hou, and Fumin Zhang. Modeling and identification of coupled translational and rotational motion of underactuated indoor miniature autonomous blimps. In *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 339–344, 2020. doi: 10.1109/ICARCV50220.2020.9305371.

Harish Venkataraman, Derya Aksaray, and Peter Seiler. Tractable reinforcement learning of signal temporal logic objectives. In *Learning for Dynamics and Control (L4DC)*, volume 120 of *Proceedings of Machine Learning Research*, pages 308–317, 2020.

Kim P. Wabersich and Melanie N. Zeilinger. Linear model predictive safety certification for learning-based control. In *IEEE Conference on Decision and Control (CDC)*, pages 7130–7135, 2018. doi: 10.1109/CDC.2018.8619829.

Kevin Zakka, Yuval Tassa, and MuJoCo Menagerie Contributors. MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo. https://github.com/google-deepmind/mujoco_menagerie, 2022.

Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, et al. Demonstrating MuJoCo playground. In *Robotics: Science and Systems (RSS)*, 2025. doi: 10.15607/RSS.2025.XXI.020.