STEM: Self-Triggered Predictive Monitoring for Dynamical Systems using Conformal Predictions [⋆]

Bowen Ye, Jianing Zhao, Junyue Huang, Shaoyuan Li and Xiang Yin [*]

[*] *School of Automation and Intelligent Sensing, Shanghai Jiao Tong University, Shanghai 200240, China.*
E-mail: {yebowen1025,jnzhao,hjy-564904993,syli,yinxiang}@sjtu.edu.cn

Abstract: Online monitoring aims to detect impending safety violations during runtime. Prior approaches typically observe the system at uniform intervals and assume a known dynamics model. We present *STEM*, a **S**elf-**T**riggered pr**E**dictive **M**onitor that is model-free and uses only historical trajectories and online scene semantics. A neural predictor, trained on logs, forecasts a short-horizon reachable region of future states. We convert this forecast into a high-confidence set via conformal prediction (CP), yielding finite-sample coverage $1 - \alpha$; to improve robustness under distribution shift, we optionally inflate the set with an f-divergence ambiguity ball. STEM couples two online decisions: a safety action (continue or stop) based on set intersection between the CP reachable set and unsafe semantics, and a self-triggered update of the next observation time dictated by an explicit risk–cost trade-off. In case studies, STEM achieves millisecond-level latency and calibrated false-alarm control at the target coverage. Compared to uniform sampling, STEM reaches the same safety level with fewer triggers. Compared to model-based reachable-set predictors, STEM requires no system model while providing $1 - \alpha$ coverage guarantees. These results indicate that adaptive sampling combined with CP-based reachable sets is an effective and practical recipe for runtime safety monitoring.

Keywords: Online monitoring; conformal prediction; f-divergence ball; self-triggered mechanism

1. INTRODUCTION

In recent years, specification-based monitoring has emerged as a promising technique for ensuring the safety and correctness of cyber-physical systems, e.g. Bartocci et al. (2018); Yan and Julius (2022); Bonnah and Hoque (2022). Compared to formal verification techniques, which exhaustively check the correctness of all possible system behaviors before execution, monitoring evaluates the correctness of the system during runtime by analyzing the actual signals generated by the system. This approach makes specification-based monitoring a lightweight and add-on module that can be integrated on top of the system. It enables the system to halt and take corrective actions when inevitable failures are detected, providing a practical and efficient way to enhance system reliability and safety. Recently, specification-based monitoring techniques have been successfully applied in many real-world cyber-physical systems such as smart cities Ma et al. (2021a), industrial IoTs Chen et al. (2020), autonomous driving Sahin et al. (2020) and power systems Beg et al. (2018).

In general, depending on the information used to evaluate the system, specification-based monitoring can be categorized into *offline monitoring* and *online monitoring*. In the offline monitoring setting, Fainekos and Pappas (2009); Donzé et al. (2013) assume that the system has executed a complete signal over the entire task horizon. As a result, the monitor can fully evaluate whether the specification is satisfied or not. In contrast, in the online monitoring setting, the system is operating in real-time, and only a partial signal up to the current instant is available to the monitor, such as in Dokhanchi et al. (2014); Deshmukh et al. (2017). Consequently, the monitor must consider the possibility of future suffix signals to make decisions, such as terminating the process if no future signal can achieve the task. This dynamic and real-time nature of online monitoring makes it particularly challenging but essential for ensuring the safety and correctness of systems during operation.

Most existing studies on online monitoring implicitly assume continuous or uniformly sampled signals. In many practical systems, however, dynamically and adaptively acquiring information at a non-uniform rate can substantially reduce energy and bandwidth consumption. For instance, an autonomous robot that is confidently operating in a safe region can sample the environment less frequently, freeing computational resources for other tasks. Along this line, Wang et al. (2024) proposes a self-triggered algorithm that adaptively determines the next observation instant for STL monitoring; yet this approach relies on an explicit system model and can be computationally demanding for online deployment, especially in high-dimensional settings.

A complementary axis concerns how to leverage system knowledge. Online signals are generated by dynamical systems that may be either black-box or explicitly modeled.

[⋆] This work was supported by the National Natural Science Foundation of China (62173226, 62061136004).

Incorporating model information can improve monitoring accuracy by ruling out dynamically infeasible future suffixes, motivating *model-based* online monitoring (e.g., STL monitoring via reachability over explicit dynamics in Yu et al. (2024)). In parallel, *model-free* predictive monitoring employs learned predictors to forecast short-horizon signals and make decisions from those forecasts, researched by Qin and Deshmukh (2020); Ma et al. (2021b); Yoon and Sankaranarayanan (2021); Momtaz et al. (2023). Our work follows the model-free line while coupling it with a self-triggered sampling rule tailored to runtime *safety* monitoring.

There is also a growing literature on applying conformal prediction (CP) to online monitoring and runtime verification. For example, Luo et al. (2024) uses CP to guarantee bounds on the false negative rate of an online monitor, and Lindemann et al. (2023); Cairoli et al. (2023) develop predictive runtime verification under STL by conformalizing trajectory prediction uncertainty. These techniques are complementary to our setting; however, prior works do not consider self-triggered, non-uniform sampling. In self-triggered monitoring, observation times depend on past data and risk, violating the exchangeability assumption underpinning standard CP and potentially causing undercoverage. To address this calibration gap, we model the trigger-induced distribution shift via an f-divergence ambiguity set mentioned in Cauchois et al. (2024) around the training distribution and conservatively inflate the CP region, yielding trigger-aware uncertainty quantification suitable for non-exchangeable sampling.

Building on these observations, we develop STEM (**S**elf-**T**riggered Pr**E**dictive **M**onitoring), a model-free, self-triggered framework for runtime *safety* monitoring. STEM trains a neural forecaster on historical trajectories to predict short-horizon behavior and calibrates these forecasts into high-confidence reachable sets via conformal prediction (CP) with finite-sample coverage $1 - \alpha$. To address the non-exchangeable sampling induced by self-triggered acquisition, we capture the trigger-induced distribution shift through an f-divergence ambiguity set and conservatively inflate the CP region. Online, STEM couples two decisions: a safety decision (continue/stop) based on set intersection between the one step CP reachable set and unsafe scene semantics, and the selection of the next observation time via a principled risk-cost trade-off. Our study restricts attention to safety specifications; the framework is instantiated accordingly and evaluated in this setting. Our contributions are threefold: (i) a model-free monitoring architecture that avoids explicit dynamics models; (ii) a CP-based calibration with f-divergence inflation that provides trigger-aware uncertainty quantification under non-uniform, observation-dependent sampling; and (iii) an online triggering rule that reduces sensing effort at maintained coverage relative to uniform sampling and compares favorably with model-based reachable-set baselines in efficiency and without reducing decision quality.

The remainder of the paper is organized as follows. In Section 2, we formalize the online monitoring problem. Section 3 introduces the necessary formal definitions, including task satisfaction and the *self-triggered monitoring mechanism.* Section 4 presents the overall framework of our approach, and Section 5 details the algorithm for solving the proposed problems. A case study is provided in Section 6 to demonstrate the applicability of the approach. Finally, Section 7 concludes the paper.

2. PROBLEM FORMULATION

In this paper, we investigate the problem of *online safety monitoring* for acting agents. The objective is to supervise the agent during execution and proactively intervene before it enters any unsafe or hazardous state.

Abstractly, we model the closed-loop interaction between the agent and its environment as an execution trace

$$\tau = (o_0, a_0, o_1, a_1, \dots),$$

where o_t denotes the observation at time t and a_t is the action chosen by the agent.

Safety requirements are specified by a property φ over (finite or infinite) prefixes of τ, which partitions system evolutions into safe and unsafe ones.

An *online monitor* is a function

$$\mathcal{M} : (o_0, a_0, \dots, o_t) \;\mapsto\; \{\texttt{continue}, \texttt{stop}\},$$

that observes the agent's behavior in real time or only knows its past trace and is allowed to override the agent by issuing a stop signal. When M outputs `stop` at time t, the agent is halted and the execution terminates.

The online monitoring problem is to design such a monitor $\mathcal{M}$ so that:

- **Preventive safety.** For every execution under M, the monitor issues `stop` strictly before the execution violates φ, thereby preventing unsafe behavior at runtime; and
- **Low intervention.** The monitor is as non-intrusive as possible, in the sense that it avoids unnecessary stop signals on executions that would remain safe without intervention.
- **Low monitoring overhead.** The monitor should retain strong safety guarantees while keeping sensing, computation, and the frequency of monitoring invocations as low as possible, for example by using a self-triggered mechanism that autonomously schedules future observation times.

Intuitively, a good monitor anticipates potential violations and intervenes just in time when danger becomes imminent, while otherwise allowing the agent to operate without interference. Its quality is thus captured by the three criteria above: preventive safety, low intervention, and low monitoring overhead. In this work, we aim to synthesize online safety monitors that satisfy these criteria by equipping them with a self-triggered mechanism that adaptively schedules future observation times.

3. PRELIMINARIES

3.1 System Model

We consider a discrete-time dynamical system with an unknown dynamic function:

$$x_{t+1} = f(x_t, w_t), \quad x_0 = \zeta,$$

where $x_t \in X \subseteq \mathbb{R}^n$ is the system state, $w_t \in W \subseteq \mathbb{R}^m$ is the disturbance with an unknown distribution, and $f : X \times$

$W \to X$ is the unknown dynamic function. The trajectory of the system over a time horizon H is a sequence of states:

$$\mathrm{x}_{0:H} = x_0 x_1 \ldots x_{H-1} x_H \in X^{H+1},$$

where $x_{t+1} = f(x_t, w_t)$ for some $w_t \in W$. Specifically, if the system monitor samples the system state consistently at each discrete time instant, then its observation of the system is a complete trajectory generated by the system.

3.2 Monitoring Specifications

In specification-based online monitoring, the objective is to assess, in real time, whether a system satisfies or violates a given specification, such as safety properties, Boolean specifications, or temporal logic formulas. In this work, we focus on the runtime monitoring problem for autonomous agents, with the aim of ensuring their operational safety. In particular, our goal is to guarantee that the agent avoids collisions and does not enter hazardous regions, even in dynamic environments where conditions may change over time.

In uniform-sampling monitoring, at each decision time t_k during operation, we predict the agent's near-future states using either (i) only the current state x_{t_k} (Markov case), or (ii) a finite history $\mathrm{x}_{t_k-L+1:t_k}$ (history-based case). Let $O_t \subset X$ denote the time-varying unsafe set (e.g., obstacles or hazardous regions extracted from the semantic map $\mathbb{M}$). A predictor $\mathcal{P}$ outputs either a conditional distribution $\hat{p}(x_{t_k+1} \mid \cdot)$ or an α-level reachable set $\widehat{\mathcal{R}}_\alpha(t_k+1) \subset \mathcal{X}$, characterizing the possible states at the next time step t_k+1. The monitoring process can be formulated as:

$$\widehat{\mathcal{R}}_\alpha(t_k+1) \cap O_{t_k+1} \neq \varnothing \implies \mathsf{stop} \quad (1a)$$

$$\widehat{\mathcal{R}}_\alpha(t_k+1) \cap O_{t_k+1} = \varnothing \implies \mathsf{continue} \quad (1b)$$

It is immediate that uniform sampling requires sensing and inference at *every* decision time t_k, incurring nontrivial energy, bandwidth, and computational costs even in benign regimes where the risk of violation is negligible. In many practical scenarios, long stretches of operation are almost surely safe with respect to the specification, rendering frequent intermediate monitoring steps redundant. These considerations motivate a self-triggered monitoring scheme that adaptively schedules the next observation time based on the predicted residual risk.

3.3 Self-Triggered Monitoring Mechanism

In general, for the purpose of energy saving, the monitor may not choose to observe the system state consistently at each predefined sampling instant. Instead, it may adaptively determine when to take the next observation sample based on online information. This leads to the notion of a self-triggered monitor, which operates as follows:

- At each decision instant t, determined in the previous decision round, the monitor observes the system state x_t by, for example, turning on the sensor.
- Based on the new observation, the monitor makes a monitoring decision in $\{0, 1\}$, where:
  - "1" denotes that 'the agent remains safe and is not expected to be in danger in the next second.", and
  - "0" denotes that "the agent is predicted to be in danger in the next second, and therefore we should stop it immediately".
- Simultaneously, along with the monitoring decision, the monitor determines an integer τ specifying after how many time instants the next observation will be made.
- The system then evolves according to its own dynamics "silently" until time instant $t+\tau+1$, at which point the above steps are repeated.

To formalize the above process, we introduce the notion of *observation history* as a sequence of the following form:

$$h = (x_0, \tau_0)(x'_1, \tau_1) \ldots (x'_n, \tau_n) x'_{n+1} \in (X \times \mathbb{N})^* X. \quad (2)$$

At this definition, each x'_i denotes the state at the i-th observation not at the i-th time instant and τ_i is the time gap between observations x'_i and x'_{i+1}. Then we denote by $\mathbb{H} := (X \times \mathbb{N})^* X$ the set of all observation histories. This leads to the following definition of self-triggered monitor.

Definition 1. (Self-Triggered Monitors). Let T_{max} be a number specifying the maximum time the monitor allowed to stay silent. A self-triggered monitor is a function:

$$\mathcal{M} : \mathbb{H} \times \mathbb{M} \to \{0, 1\} \times \{0, 1, \ldots, T_{max}\} \quad (3)$$

that makes decisions based on the observation histories. $\mathbb{M}$ means semantic map, mapping from world coordinates to semantic labels. Specifically, if the most recent observation does not violate the prescribed safety rules, the monitor selects an inter-sample interval and schedules the next observation time; if a violation is detected, it intervenes (e.g., issues a stop). This yields runtime safety with adaptive (non-uniform) sampling and reduced sensing or compute load.

4. SELF-TRIGGERED PREDICTIVE MONITORING

4.1 Overall Framework

Before formally presenting how to construct high-quality monitors that both ensure safety and reduce the frequency of monitoring, we first introduce a methodological scaffold via an architectural decomposition of our CP-based STEM framework. Specifically, our approach consists of the following three components:

- **Point Predictor:** The monitoring process requires predictive capabilities for future state estimation, which is challenged by two key factors: (i) the system dynamics and environmental disturbances are unknown, and (ii) the presence of closed-loop control policies. To address these challenges, a neural network–based deterministic point predictor g_θ is employed to provide consistent online forecasts under uncertainty.
- **Trajectory generator:** Under the self-triggered mechanism, the observed information becomes temporally discontinuous, whereas the training data are collected at uniform sampling intervals. As a result, test-time observations cannot be fed directly into the neural network trained on continuous trajectories. A naive attempt to align the two by subsampling the training data (e.g., via point deletion) would require test-specific architectural adaptations and effectively lead to a combinatorial explosion in model configurations and parameters, substantially increasing both

complexity and computational cost. To avoid this, we introduce a trajectory generator I_θ. Its input is the discontinuous observation history, and its output is a continuous trajectory obtained via linear interpolation. The reconstructed trajectory produced by I_θ is then used as the input to the point predictor g_θ.

- **Conformal Prediction with F-divergence ball:** Then we use conformal prediction P_θ to extend deterministic point prediction into uncertainty-calibrated prediction regions with *confidence guarantees*. However, the synthesized trajectories violate the i.i.d. assumption required for training data and testing data. To address this distribution shift between training and test data and quantify the correct probability of prediction regions, we propose a f-divergence ball framework in Section 4.5.

Next, we provide the technical details of each part.

4.2 Point Predictor

We employ a neural network–based deterministic point predictor to estimate the next state of the trajectory given its recent history. Let the observed trajectory segment at time t be denoted as

$$\mathrm{x}_{t-L+1:t} = x_{t-L+1}x_{t-L+2}\dots x_t \in X^L,$$

where L is the fixed observation length and $x_t \in \mathbb{R}^d$ represents the position (e.g., $d = 2$ for (x, y)). The point predictor $g_\theta(\cdot)$, parameterized by neural network weights θ, produces the one-step prediction

$$\hat{x}_{t+1} = g_\theta(\mathrm{x}_{t-L+1:t}),$$

which corresponds to the expected next state under current motion patterns. The model is trained by minimizing the mean squared prediction error:

$$\mathcal{L}_{\text{pred}}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left\|x_{t+1}^{(i)} - g_\theta(\mathrm{x}_{t-L+1:t}^{(i)})\right\|_2^2,$$

where N is the number of training samples. This deterministic predictor serves as the foundation for constructing the conformal prediction regions introduced in Section 4.4, providing a nominal trajectory around which symmetric uncertainty sets are later formed.

4.3 Trajectory Generator

Given two endpoints x'_t and x'_{t+1} of a trajectory and the time interval τ_t between them. We use linear interpolation to generate points between two endpoints, which can be formulated as:

$$x_{t,i} = TG(x'_t, x'_{t+1}, \tau_t) = (1-\lambda_t)\times x'_t + \lambda_t \times x'_{t+1} \quad (4)$$

where $i \in I = \{1, \dots, \tau_t\}$ and λ_t is determined by time interval τ_t.

Remark 1. We adopt linear interpolation in trajectory synthesis primarily to control and quantify distribution shift introduced by reconstruction. Concretely, given an original trajectory, we delete several intermediate samples and then reconstruct the trajectory via a linear interpolation operator. Because the operator is linear and preserves endpoints, the discrepancy between the reconstructed and original trajectory, which is measured at the sample level or in distributional metrics, admits closed-form characterization or tight bounds. This yields an analytically tractable "distribution law" for the synthesized data and allows us to calibrate, monitor, and limit the shift induced by resampling and reconstruction. The details can be found in Luor (2018).

4.4 Conformal Prediction

Conformal prediction (CP) is used to quantify the uncertainty of traditional supervised learning. It is a very general approach that can be applied across all existing deterministic classifiers and regressors Balasubramanian et al. (2014). CP produces prediction regions with guaranteed probability by enriching point-wise predictions with guaranteed validity.

Definition 2. (Conformal Prediction Regions). Assume that the significance level $\alpha \in (0, 1)$ and the test input is x^* with its true corresponding target y^*. The goal of CP is to construct the α-prediction region for x^*, $\Gamma_*^\alpha \subseteq Y$ that satisfies:

$$\mathbb{P}_{(x^*,y^*)\sim Z}(y^* \in \Gamma_*^\alpha) \geq 1-\alpha \quad (5)$$

It shows that prediction region is guaranteed to contain the true (unknown) value y^* with confidence $1-\alpha$.

The idea of CP is to construct the prediction region by "remove" areas that do not meet the requirements. Firstly for $R_1, R_2, \dots, R_{n+1}$ be $n+1$ independent and identically distributed (i.i.d) random variables. The goal in conformal prediction is to obtained a prediction region for R_{n+1} based on $R_1, R_2, \dots, R_n$. Formally expressed in (5), given a significance level $\alpha \in (0, 1)$, we want to obtain a prediction region $R_{n+1} \leq C$ such that

$$\mathbb{P}(R_{n+1} \leq C) \geq 1-\alpha \quad (6)$$

We refer to R_i as nonconformity score (NCF). For supervised learning, we can select $R_i := \|y_i - g_\theta(x_i)\|$ where g_θ is a point predictor with a network structure of RNN or LSTM and $\|y_i - g_\theta(x_i)\|$ is score function $s(x_i, y_i)$ so that a large nonconformity score indicates a poor predictive model. Refer to Tibshirani et al. (2019), we add $R_\infty = \infty$ to $R_1, \dots, R_n$ and sort them in non-decreasing order ($\bar{R}$ refers to the order static of R). So for $\bar{R}$, we have $\bar{R}_{i+1} \geq \bar{R}_i$. By setting $q := \lceil (n+1)(1-\alpha)\rceil \leq n$, we obtain the $(1-\alpha)$th quantile as $C := \bar{R}_q$. So we can get the prediction region $\Gamma_*^\alpha \subseteq Y$ for x^*:

$$\Gamma_*^\alpha := [y - C, y + C] \quad (7)$$

where $y = g(x^*)$ and C is the $(1-\alpha)$th quantile in calibration set Z_c.

4.5 Distributions Shift over F-divergence Balls

Since the generated trajectories and training set are definitely not i.i.d or exchangeable, we introduce f-divergence ball here. In Cauchois et al. (2024), the author propose Robust Validation, which can be used when distributions shift. In our prediction process, score function is $s(x, y) = \|y - g_\theta(x)\|$ where g is point predictor defined is Section 4.2. We firstly define the f-divergence Csiszár (1967) between two probability distributions Q and Q' on a set Z is:

$$D_f(Q\|Q') := \int_{z\in Z} f\left(\frac{dQ(z)}{dQ'(z)}\right)dQ'(z) \quad (8)$$

f is a closed convex function $f : R \to R$ satisfying $f(1) = 0$ and $f(t) = +\infty$ for $t < 0$. A f-divergence ball with radius ρ and initial distribution Q is:

$$B_{f,\rho}(Q) = \{Q' : D_f(Q'||Q) \leq \rho\} \tag{9}$$

We can valid the prediction model if the testing data distribution Q_{test} is on the f-divergence ball constructed by Q_{train}.

We present our expanded prediction region and provide mathematical probabilistic guarantees that the region conforms to the actual range. The correct prediction set is:

$$C^{corr}_{f,\rho}(x) := \{y \in Y : s(x,y) \leq Quantile_{f,\rho}(1-\alpha; Q_{train})\} \tag{10}$$

where Y is the state space, $s(x,y)$ is the score function, α is significance level set by us and $Quantile$ is defined as:

$$Quantile_{f,\rho}(\alpha; Q) = \sup_{Q' \in B_{f,\rho}(Q)} Quantile(\alpha; Q') \tag{11}$$

We also need to set a value for ρ, which is shows below:

$$\rho \geq \rho^* = D_f(Q_{test}||Q_{train}) \tag{12}$$

Now we can provide the probability guarantee of the actual location in prediction area:

$$\mathbb{P}(Y_{n+1} \in C^{corr}_{f,\rho}(X_{n+1})) \geq 1-\alpha \tag{13}$$

Here X_{n+1} means the continuous past information and Y_{n+1} means the predicted future states. The predictor function P_θ whose input is continuous state information and prediction regions Γ is:

$$\begin{aligned}\Gamma &= P_\theta(X_{n+1}) = C^{\text{corr}}_{f,\rho}(X_{n+1}) \\ &= \{y \in Y : ||y - g_\theta(x)|| \leq Quantile_{f,\rho}(1-\alpha; Q_{train})\} \\ &= R(g_\theta(x), f, \rho, \alpha, Q_{train}) \end{aligned} \tag{14}$$

5. SOLVING ALGORITHM

In this section, we will propose our solving algorithm for self-triggered online monitoring problem with unknown model.

We first describe how to compute the distribution Q using the predictor g_θ and the dataset, as summarized in Algorithm 1. Specifically, we begin by computing the nonconformity score for each sample in the dataset and then form the empirical distribution of these scores. Using Equation 4.5, we subsequently obtain the expansion parameter ρ. We denote this procedure by $CD(Z_{\text{ref}}, Z_{\text{shift}}, g_\theta, s)$.

Then we train a deterministic point predictor g_θ that maps a fixed-length history of L past states to the next-step position. The training process is shown in Algorithm 2.

Algorithm 1: Calculating distribution Q for two disjoint calibration sets

```
Input: reference data $Z_{ref}$, shift data $Z_{shift}$, point
       predictor $g_\theta$ and score function $s$
Output: Expansion parameter $\rho$
for $k \in \{ref, shift\}$ do
    Set $S_k = \emptyset$
    for $i = \{1, \ldots, |Z_k|\}$ do
        $r_i^k = s(x_i^k, y_i^k) = ||y_i^k - g_\theta(x_i^k)||$
        $S_k = S_k \bigcup \{r_i^k\}$
    $\widehat{Q_k}(t) = \frac{1}{|S_k|}\sum_{i=1}^{|S_k|} 1\{r_i^k \leq t\}, r_i^k \in S_k$
$\rho = D_f(\widehat{Q_{shift}}||\widehat{Q_{ref}})$
Return Expansion parameter $\rho$
```

Algorithm 2: Training for P_θ^{NU} with CP

```
Input: Trajectory data $Z$, score function $s$ and a
       Pre-set function $f$
Output: Expansion parameter $\rho$ and point predictor
        $g_\theta$
Split $Z$ into $Z_{train}$, $Z_{cal-ref}$ and $Z_{cal-shift}$
Using $Z_{train}$ to train a point predictor $g$ with input
 length $L$
Construct $Z'_{cal-shift} = \emptyset$
for $(x,y) \in Z_{cal-shift}$ do
    delete some points in $x$ to get $x'$
    $Z'_{cal} = Z'_{cal} \bigcup \{(x', y)\}$
Construct a set $S = \emptyset$
for $(x', y) \in Z'_{cal-shift}$ do
    new continuous path $x'' = I_\theta(x')$
    $S = S \bigcup \{(x'', y)\}$
$\rho = CD(Z_{cal-ref}, Z_{cal-shift}, g_\theta, s)$
Return Expansion parameter $\rho$, point prediction $g_\theta$
```

Remark 2. In Algorithm 2, we first partition the calibration set into two splits. On one split, we randomly delete a subset of points along each trajectory and use the trajectory generator I_θ to synthesize the missing segments, producing complete trajectories. The resulting sample is constructed to match the test-time distribution; we therefore treat it as a surrogate test set and use it to calibrate the f-divergence ball for distribution-free robustness.

Algorithm 3 presents the online monitoring routine of **STEM**, which schedules re-triggering based on conformal uncertainty propagation. Given the most recent observation history h, the trained point predictor g produces a one-step forecast and its conformal prediction region R'_{t+1}. The algorithm then rolls this region forward over a planning horizon, repeatedly (i) extracting obstacle sets from the online semantic map $M'_{t+\tau}$, (ii) sampling support points on the boundary $\partial R'_{t+\tau}$ to capture worst-case propagation, (iii) pushing each support point through g to obtain the next-step regions, and (iv) merging and closing these regions to form a single reachable set for the next step. We declare an immediate alarm iff $\tau = 0 (s = 0)$; otherwise we certify a sleeping interval of length τ and set $s = 1$, which means continue operating in "sleeping". This region-to-region rollout preserves a conservative safety guarantee while avoiding unnecessary, overly frequent monitoring.

6. CASE STUDY

In this section, we provide some motivating examples to show that our STEM can not only guarantee safety during online monitoring , but also cost less than time-triggered mechanism. The provided examples cover a variety of scenarios in which an autonomous agent operates under different environmental conditions. In each case, we employ **STEM** as the runtime monitor to adaptively determine safe re-evaluation intervals and ensure reliable operation.

6.1 Dynamic Model

The dynamic system model function f for experiment is unknown to us, neither do disturbance noise. However,

Algorithm 3: Online Monitoring Mechanism using STEM

```
Input: past observation $h$, trained point predictor $g$,
       expansion radius $\rho$, semantic map $M'_t$ at each
       step, max "sleeping" interval $T$, pre-set
       function $f$, significance level $\alpha$, training
       dataset distribution $Q_{train}$
Output: "sleeping" interval $\tau$ and monitoring
        decision $\in \{0, 1\}$
Set $\tau = T$ and $s = 1$
Select a sequence $x'_{t-L} : x'_t$ from h
continuous trajectory $\mathrm{x_{t-L:t}} = TG(x'_{t-L} : x'_t)$
next possible point $y_{t+1} = g(\mathrm{x_{t-L:t}})$
next region prediction is
 $R'_{t+1} = R(y_{t+1}, f, \rho, \alpha, Q_{train})$
for $\tau_0 \in \{0, \ldots, T\}$ do
    Extract obstacle region $O'_{t+\tau_0}$ from $M'_t$
    if $R'_{t+\tau_0+1} \bigcap O'_t \neq \emptyset$ then
        Set $\tau = \tau_0$
        break
    else
        Set $\tau_0 = \tau_0 + 1$
    Get the closed boundary $\partial R$ from $R'_{t+1}$
    Select $N$ points $\{x^i_{t+1}\}^N_{i=1}$ from $\partial R$
    Set new prediction region as $R'_{t+\tau_0+1} = \emptyset$
    for $x \in \{x^i_{t+1}\}^N_{i=1}$ do
        Get new trajectory $\pi$ by delete the first point
         in $\pi'$ and added $x$ into it
        next possible point $y_{t+\tau_0+1} = g(\pi)$
        next region prediction is
         $R_{t+\tau_0+1} = R(y_{t+\tau_0+1}, f, \rho, \alpha, Q_{train})$
        $R'_{t+\tau_0+1} = R'_{t+\tau_0+1} \bigcup R'_{t+\tau_0+1}$
    Get the closure $cl(R'_{t+\tau_0+1})$ as the next step
     prediction region $R'_{t+\tau_0+1}$
if $\tau = 0$ then
    Set $s = 0$
Return $\tau$ and decision $s$
```

we have access to a dataset Z of the agent's operating trajectories.

6.2 Implementation details

We train an LSTM network as the point predictor g with a learning rate of 5×10^{-5}, 100 training epochs, and a batch size of 64. The prediction interval is constructed at significance level $\alpha = 0.01$, corresponding to approximately 99% marginal coverage for the next-step location. All experiments are conducted on a single NVIDIA RTX 4090 (24 GB VRAM).

6.3 Experiments and Results

In this subsection, we present six representative snapshots of our running results. They cover two types of scenarios: one with simple obstacle configurations and another with more challenging obstacle layouts. In all figures, the black region around the perimeter denotes the boundary of the workspace, which is impassable. The gray areas in the interior correspond to obstacles and represent unsafe regions. A single agent moves within this environment: the blue dot marks the initial position, and the blue

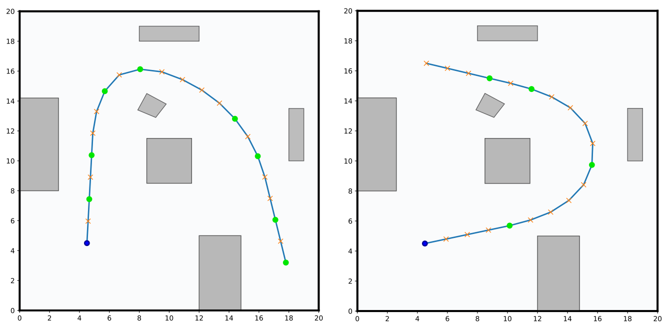
Fig. 1. Scenario with simple obstacles.

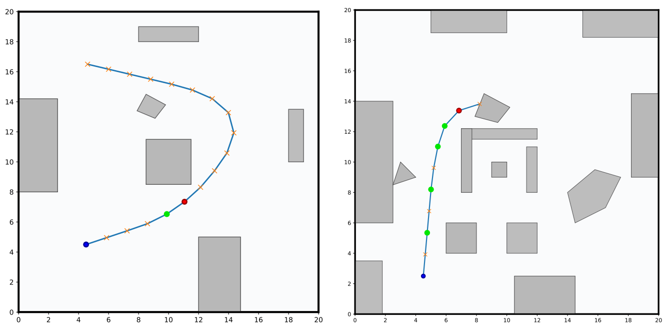
Fig. 2. Scenario with stop signals.

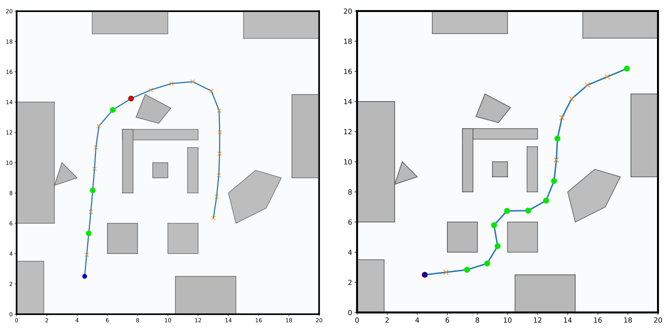
Fig. 3. Scenario with complex obstacles.

curve denotes the nominal trajectory that the agent would follow without any monitoring or intervention. The orange dots indicate supervision points when a fixed supervision interval of 1 is used. In contrast, the green dots denote the supervision points selected by **STEM**, and the red dots mark the time instants at which STEM decides to stop the agent.

From these results, we observe that the STEM algorithm not only reduces the number of supervision events and the associated monitoring cost, but also maintains safety by stopping the agent before an imminent or high–risk collision occurs. Moreover, STEM adapts its behavior to the risk level: when the agent is close to obstacles, the supervision becomes more frequent (the algorithm "opens its eyes" more often) to avoid collisions, whereas when the agent is far from any obstacle, the supervision frequency is reduced, thereby lowering computational and sensing overhead.

7. CONCLUSION

This paper studies runtime safety monitoring and introduces STEM, a self-triggered, model-agnostic predictive monitor that maximizes sensor sleep while certifying safety. To mitigate distribution shift induced by self-triggered sampling between reconstructed trajectories and the training data, we further incorporate an f-divergence–based correction to calibrate predictions under shift. Although our focus is safety, the framework is generic and amenable to broader online monitoring tasks. Future work will extend STEM to more general scenes, e.g. specification-aware monitoring, multi-agent settings, and tighter adaptive calibration.

DECLARATION OF GENERATIVE AI AND AI-ASSISTED TECHNOLOGIES IN THE WRITING PROCESS

During the preparation of this work, we used ChatGPT to assist with polishing and editing parts of the manuscript. After using this tool, we carefully reviewed and revised all generated content as needed and take full responsibility for the final version of the paper.

REFERENCES

Balasubramanian, V., Ho, S.S., and Vovk, V. (2014). *Conformal prediction for reliable machine learning: theory, adaptations and applications*. Newnes.

Bartocci, E., Deshmukh, J., Donzé, A., Fainekos, G., Maler, O., Ničković, D., and Sankaranarayanan, S. (2018). Specification-based monitoring of cyber-physical systems: a survey on theory, tools and applications. *Lectures on Runtime Verification: Introductory and Advanced Topics*, 135–175.

Beg, O.A., Nguyen, L.V., Johnson, T.T., and Davoudi, A. (2018). Signal temporal logic-based attack detection in dc microgrids. *IEEE Transactions on Smart Grid*, 10(4), 3585–3595.

Bonnah, E. and Hoque, K.A. (2022). Runtime monitoring of time window temporal logic. *IEEE Robotics and Automation Letters*, 7(3), 5888–5895.

Cairoli, F., Paoletti, N., and Bortolussi, L. (2023). Conformal quantitative predictive monitoring of stl requirements for stochastic processes. In *Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control*, 1–11.

Cauchois, M., Gupta, S., Ali, A., and Duchi, J.C. (2024). Robust validation: Confident predictions even when distributions shift. *Journal of the American Statistical Association*, 119(548), 3033–3044. doi:10.1080/01621459.2023.2298037. URL http://dx.doi.org/10.1080/01621459.2023.2298037.

Chen, G., Liu, M., and Kong, Z. (2020). Temporal-logic-based semantic fault diagnosis with time-series data from industrial internet of things. *IEEE Transactions on Industrial Electronics*, 68(5), 4393–4403.

Csiszár, I. (1967). On information-type measure of difference of probability distributions and indirect observations. *Studia Sci. Math. Hungar.*, 2, 299–318.

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

Dokhanchi, A., Hoxha, B., and Fainekos, G. (2014). Online monitoring for temporal logic robustness. In *International Conference on Runtime Verification*, 231–246. Springer.

Donzé, A., Ferrere, T., and Maler, O. (2013). Efficient robust monitoring for stl. In *International conference on computer aided verification*, 264–279. Springer.

Fainekos, G.E. and Pappas, G.J. (2009). Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42), 4262–4291.

Lindemann, L., Qin, X., Deshmukh, J.V., and Pappas, G.J. (2023). Conformal prediction for stl runtime verification. In *Proceedings of the ACM/IEEE 14th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2023)*, 142–153.

Luo, R., Zhao, S., Kuck, J., Ivanovic, B., Savarese, S., Schmerling, E., and Pavone, M. (2024). Sample-efficient safety assurances using conformal prediction. *The International Journal of Robotics Research*, 43(9), 1409–1424.

Luor, D.C. (2018). Statistical properties of linear fractal interpolation functions for random data sets. *Fractals*, 26(01), 1850009.

Ma, M., Bartocci, E., Lifland, E., Stankovic, J.A., and Feng, L. (2021a). A novel spatial–temporal specification-based monitoring system for smart cities. *IEEE Internet of Things Journal*, 8(15), 11793–11806.

Ma, M., Stankovic, J., Bartocci, E., and Feng, L. (2021b). Predictive monitoring with logic-calibrated uncertainty for cyber-physical systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 20(5s), 1–25.

Momtaz, A., Basnet, N., Abbas, H., and Bonakdarpour, B. (2023). Predicate monitoring in distributed cyber-physical systems. *International Journal on Software Tools for Technology Transfer*, 25(4), 541–556.

Qin, X. and Deshmukh, J.V. (2020). Clairvoyant monitoring for signal temporal logic. In *Formal Modeling and Analysis of Timed Systems: 18th International Conference, FORMATS 2020, Vienna, Austria, September 1–3, 2020, Proceedings 18*, 178–195. Springer.

Sahin, Y.E., Quirynen, R., and Di Cairano, S. (2020). Autonomous vehicle decision-making and monitoring based on signal temporal logic and mixed-integer programming. In *2020 American Control Conference (ACC)*, 454–459. IEEE.

Tibshirani, R.J., Foygel Barber, R., Candes, E., and Ramdas, A. (2019). Conformal prediction under covariate shift. *Advances in neural information processing systems*, 32.

Wang, C., Yu, X., Zhao, J., Lindemann, L., and Yin, X. (2024). Sleep when everything looks fine: Self-triggered monitoring for signal temporal logic tasks. *IEEE Robotics and Automation Letters*.

Yan, R. and Julius, A. (2022). Distributed consensus-based online monitoring of robot swarms with temporal logic specifications. *IEEE Robotics and Automation Letters*, 7(4), 9413–9420.

Yoon, H. and Sankaranarayanan, S. (2021). Predictive runtime monitoring for mobile robots using logic-based bayesian intent inference. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 8565–8571. IEEE.

Yu, X., Dong, W., Li, S., and Yin, X. (2024). Model predictive monitoring of dynamical systems for signal temporal logic specifications. *Automatica*, 160, 111445.