Adaptive splitting of reusable temporal monitors for rare traffic violations

Citation for published version:

Digital Object Identifier (DOI):
10.48550/arXiv.2405.15771

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Proceedings of the 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Adaptive Splitting of Reusable Temporal Monitors for Rare Traffic Violations

Craig Innes and Subramanian Ramamoorthy

Abstract—Autonomous Vehicles (AVs) are often tested in simulation to estimate the probability they will violate safety specifications. Two common issues arise when using existing techniques to produce this estimation: If violations occur rarely, simple Monte-Carlo sampling techniques can fail to produce efficient estimates; if simulation horizons are too long, importance sampling techniques (which learn proposal distributions from past simulations) can fail to converge. This paper addresses both issues by interleaving rare-event sampling techniques with online specification monitoring algorithms. We use adaptive multi-level splitting to decompose simulations into partial trajectories, then calculate the distance of those partial trajectories to failure by leveraging robustness metrics from Signal Temporal Logic (STL). By caching those partial robustness metric values, we can efficiently re-use computations across multiple sampling stages. Our experiments on an interstate lane-change scenario show our method is viable for testing simulated AV-pipelines, efficiently estimating failure probabilities for STL specifications based on real traffic rules. We produce better estimates than Monte-Carlo and importance sampling in fewer simulations.

I. STATISTICAL SIMULATION FOR AV TESTING

Autonomous Vehicles (AVs) typically undergo rigorous simulated testing before deployment [36]. A standard set of steps for testing is as follows: First we define a scenario (e.g., a highway lane-change expressed in OpenScenario [39]). Next, we define a safety specification (e.g., “avoid impeding traffic flow”) in a formal language like Signal Temporal Logic (STL). Then, we run stochastic simulations to estimate the probability our AV-system violates our specification [11]. This statistical simulation approach is used because modern AVs contain “black box” components like Neural-Network perception modules and non-linear solvers. Such components provide few analytical guarantees over their behaviour.

A core problem plaguing statistical simulation is estimating rare events. Consider a stochastic simulation scenario where there exists a $10^{-4}$ probability that random noise in the sensors will cause our AV to “fail” (i.e., to violate our safety specification). If we ran 100 simulations, it is likely none would produce a failure. Even if sampling did produce a failure, estimation variance would be unacceptable [23].

Many works address rare-event problems for AVs via Importance Sampling [6]: Importance samplers draw simulations from a proposal distribution where the factors leading to a failure occur more frequently. The final estimate of failure probability is then re-weighted to reflect the original distribution. Since we do not know in advance all combinations of states which result in failure, such techniques must learn a good proposal. This learning step has no convergence guarantees, and probability estimates from such adaptive techniques can have unbounded error [3]. Importance sampling also tends to fare better when failures are caused by instantaneous single-state errors, but in the AV domain, failures often occur as a result of accumulated errors over dependent states [9].

This paper instead proposes an approach to AV rare-event simulation based on merging Adaptive Multi-level Splitting (AMS) [9] with STL monitoring. AMS relies on estimating probabilities for a sequence of decreasing failure thresholds $\gamma_1 > \gamma_2 > \cdots > \gamma_M$, where the final $\gamma_M$ is equivalent to the rare failure event of interest. The key idea is that estimating any intermediate $\gamma_i$ (given $\gamma_{i-1}$) is easier than estimating $\gamma_M$ outright. To adapt AMS from estimating isolated phenomena (e.g., particle transport [27]) to estimating complex AV-system failures, we face two issues:

The main issue is how to consistently produce simulations which fall below those intermediate failure thresholds $\gamma_0 \ldots \gamma_M$, and how to efficiently measure the distance to failure in the first place. Our approach measures failure using metrics for evaluating STL specification robustness. By leveraging online monitoring [13], we can cache the robustness values of partial trajectories, stop simulations at the point where they fall below the current threshold, and re-sample from this point onwards to produce trajectories which fall below subsequent failure thresholds.

A secondary issue is generating stochastic AV perceptual errors. Approaches which assume noise follows a well-known (e.g., Gaussian) state-independent distribution [7] are insufficient to capture the perceptual variety of a typical AV-system—a LiDAR detector may be great for close range traffic, but terrible for long range or occluded traffic [33]. We therefore use a Perception Error Model (PEM) [34]--a...
surrogate trained on real sensor data which mimics perceptual errors encountered in regular operation (Sec II-A).

The main contribution of this paper is a new method for assessing failure probability in AV-scenarios (Sec III), combining AMS (Sec III-B), PEMS (Sec II-A), and online STL monitoring (Sec II-C). Our experiments focus on the case study of a highway lane-change scenario, and show our method can be used to test a full AV-pipeline—perception down to control (Sec IV). Our approach outperforms Monte-Carlo sampling, as well as fixed and adaptive importance sampling, across various STL specifications (Sec IV-A).

To limit the scope of experiments, this paper exclusively considers probabilistic noise in the perception system as the primary source of simulation stochasticity (As is standard in other works [10]). However, our proposed sampling method can easily be applied to simulators which consider other sources of stochasticity such as those arising from traffic behaviour or physical uncertainties.

II. PROBABILITY OF FAILURE IN BLACK BOX SIMULATION

Consider the lane-change maneuver in Fig 1. Our car (the left-most ego vehicle), must change to the left lane to avoid an obstacle, then re-merge. Formally, let’s assume our scenario takes place over a total of $T$ time steps. We denote the $d$-dimensional state of our ego vehicle at time $t$ as $x^{ego}_t \in \mathbb{R}^d$; other vehicles as $x^{oi}_t$. For succinctness, we write $x_t = \{x^{ego}_t, x^{oi}_t, \ldots, x^{odi}_t\}$ for the combined state. The state $x_t$ contains the position, velocity and rotation of each vehicle. At each time step $t$, the ego vehicle’s control system takes an action $a_t \in \mathbb{R}^2$ (desired acceleration and turn-velocity) with the aim of minimizing costs associated with competing driving goals (e.g., maintaining a reference velocity and minimizing abrupt steering), and subject to constraints (e.g., limits on acceleration, avoiding collisions, staying within road boundaries). We can run a simulation of this system to generate a trajectory $\pi = [(x_0, a_0) \ldots (x_T, a_T)]$, where $\pi_{[i:j]} = [(x_i, a_i) \ldots (x_j, a_j)]$ denotes a partial slice. For a given scenario, we wish to test whether our above AV-system will violate an STL safety specification $\varphi$. Due to probabilistic noise in the perception system, our simulator is inherently stochastic. Therefore our aim is to calculate the probability that, for a random run of our simulator, our AV-system will violate $\varphi$:

$$P_{fail} = \mathbb{E} \left[ \mathbb{I} \{ \pi \not\models \varphi \} \right]$$

(1)

where $\mathbb{I} \{ \pi \not\models \varphi \}$ is an indicator function which returns 1 if $\pi$ violates $\varphi$ and 0 otherwise. To explain how our method efficiently calculates (1), we first cover the pre-requisites for perception, tracking, and control for simulating our AV (Sec II-A II-B). We then cover defining safety specifications $\varphi$ in STL, and how to quantify their satisfaction using a robustness metric (II-A). We can then describe our main contribution—interleaving online monitoring and Adaptive Multi-level Splitting to estimate a failure probability for AV-systems via statistical simulation (II-B).

A. Simulated State Estimation with PEMS

Most AV problems assume our system does not have access to the true state $x_t$. Instead our AV must estimate this state via observations from its sensors (see Fig 2). At time $t$, let us denote a full-snapshot of the world by $w_t$. This snapshot implicitly contains the relevant ground-truth state $x_t$, but also other scene information (e.g., vehicle types, dimensions etc.). A sensor $S$ can be thought of as a function which takes $w_t$ and produces high-dimensional raw sensor data $S(w_t)$ (e.g., a LiDAR point-cloud). This sensor data is then passed to a perception function $f$ (e.g., a neural-network obstacle detector [35]), to produce an observation $y_t \in \mathbb{R}^{d'}$ (e.g., the bounding boxes of other vehicles relative to the ego):

$$y_t = f(S(w_t))$$

(2)

By using standard tracking algorithms [42], we can use these observations to get an estimate $\hat{x}_t$ of the current state:

$$\hat{x}_t = \mathbb{E} [x_t | y_{0..t}]$$

(3)

If we have a one-step vehicle dynamics model $f_{dyn}$, we can use it to predict the state in future time steps:

$$\hat{x}_{t+i|t} = f_{dyn} \circ \cdots \circ f_{dyn}(\hat{x}_t)$$

(4)

Here, $\hat{x}_{t+i|t}$ denotes the predicted state at time $t+i$ given an estimate at $t$. In a real-world system, this setup allows us to sense, estimate, track and predict the state; in simulation, we have a problem: $f$ is typically a data-driven perception module trained on real sensors, but most simulators cannot generate high fidelity sensor inputs (e.g., photorealistic images). We can resolve this issue by re-framing our perception system as a noisy projection from the state space to observation space:

$$y_t = f(S(w_t)) = H x_t + \epsilon(g(w_t))$$

(5)

In this view, $f$ is composed of a $d' \times d$ projection matrix $H$ on state $x_t$, plus stochastic error dependent on the current world state $w_t$. The $\epsilon$ function is a surrogate model known as a Perception Error Model [34]. This is a probabilistic model of the original AV’s perception noise, dependent on salient features $g(w)$ extractable from simulated $w$. Salient features can include obstacle positions, dimensions, occlusion, or
environment factors. We model $\epsilon$ as a gaussian process [43], where $m$, $\kappa$ are mean/kernel functions.

$$\epsilon(w) \sim \text{GP}(m(g(w)), \kappa(g(w), g(w')))$$ (6)

Now, instead of using real-world sensor inputs directly, our simulator applies probabilistic noise from the PEM based on the current simulated state. This makes each run of the simulator inherently stochastic, as different amounts of perceptual noise may be applied to observations on each run.

B. Model Predictive Control for Highway Maneuvers

For a sufficiently complex control task such as lane changing, choosing the best actions $a_t$ at each time step is a non-linear constrained control task. We phrase the controller of our AV system under test as a Receding-Horizon Model-Predictive Control optimization [38]. Eq (7) provides a formal definition of the optimization problem: At each time step $t$, the controller aims to choose actions $a_{t:(t+H)}$ over a finite time horizon $H$ which minimize a cost function $J(x,a)$. Actions must be chosen subject to a set of constraints $c_j(x,a)$ and obey physical dynamics $f_{dyn}$:

$$\min_{x_{t:(t+H)}, a_{t:(t+H)}} \sum_{k=t}^{t+H} J(x_k, a_k)$$

$$\text{s.t.}$$

$$x_t = \hat{x}_t$$

$$\forall k, x_{k+1} = f_{dyn}(x_k, a_k)$$

$$\forall j, c_j(x_k, a_k) < 0$$ (7)

Cost function $J(x,a)$ balances multiple factors such as tracking a reference velocity, minimizing abrupt movement, and staying close to the lane centre. The constraint functions $c_j(x,a)$ ensure states and actions remain feasible (e.g., that the car stays within road bounds and acceleration limits).

We give a further breakdown of the cost function and implementation in Section IV. However, the purpose of describing Eq (7) here in the context of our testing problem is to highlight that our AV-controller represents yet another “black-box” component of our system: Despite behaving deterministically, solvers for nonlinear control problems are not guaranteed to find a global solution, and can perform arbitrarily poorly [18]. Other typical methods (such as reinforcement learning), pose the same problem.

III. ESTIMATING SPECIFICATION FAILURE PROBABILITY

We have defined our testing goal and outlined the components of our system-under-test. Now we can show how we formalize our safety properties, how we draw samples from the simulator, and how those aspects interact.

A. Specifying Safety with Signal Temporal Logic

We can express AV traffic rules involving statements about continuous values over time using Signal Temporal Logic [30]. STL has grammar:

$$\varphi := T | \eta | \neg \varphi | \varphi_1 \land \varphi_2 | \Box_I \varphi |$$

$$\Diamond_I \varphi | \varphi_1 \land \varphi_2 | H_I \varphi | O_I \varphi$$ (8)

Here, $\eta$ is any predicate $\rho(x) - b \leq 0$ (with $b \in \mathbb{R}$ and function $\rho$ from state $x$ to $\mathbb{R}$). $\Box_I \varphi$ means $\varphi$ is always true at every future time in interval $I$. $\Diamond_I \varphi$ means $\varphi$ must eventually be true in $I$, $\varphi_1 \land \varphi_2$ means $\varphi_1$ must remain true within $I$ until $\varphi_2$ is true. $H_I$, $O_I$ are past versions of $\Box_I$, $\Diamond_I$.

We can convert $\tau$ and $\varphi$ to a robustness metric $\mathcal{L} (\varphi, \tau, i)$ over trajectories. This measures how strongly $\tau$ satisfied $\varphi$ (starting from time $i$). Large positive values indicate robust satisfaction; negative values a strong violation; near-zero values a trajectory on the boundary of satisfaction/violation. Eq (9) shows a subset of $\mathcal{L}$’s semantics. Other operators follow similar definitions [13].

$$\mathcal{L}(\rho(\tau) > 0, \tau, i) = \rho(x_i)$$

$$\mathcal{L}(\neg \varphi, \tau, i) = -\mathcal{L}(\varphi, \tau, i)$$

$$\mathcal{L}(\varphi_1 \land \varphi_2, \tau, i) = \min\{\mathcal{L}(\varphi_1, \tau, i), \mathcal{L}(\varphi_2, \tau, i)\}$$

$$\mathcal{L}(\Box_I \varphi, \tau, i) = \inf_{i' \in i+I} \mathcal{L}(\varphi, \tau, i')$$ (9)

B. Estimating the Rare Event with Splitting

With our PEM, AV-controller, specification $\varphi$ and metric $\mathcal{L}$, we now describe our adaptive sampling contribution: Given $N$ simulated trajectories $\tau^{(1..N)}$, we wish to calculate Eq (1)—the probability a simulation violates $\varphi$. A naive approach might use Monte-Carlo sampling:

$$\mathbb{E}[\mathcal{I}\{\mathcal{L}(\varphi, \tau, 0) < \gamma\}] \approx \frac{1}{N} \sum_{i=1}^{N} \mathcal{I}\{\mathcal{L}(\varphi, \tau, 0) < \gamma\}$$ (10)

However, when violating $\varphi$ is rare, the number of simulations needed to achieve low relative error rapidly becomes infeasible [23]. We instead take a multi-level splitting approach [9]. Rather than immediately estimating failure ($\gamma = 0$) we instead estimate decreasing thresholds $\gamma_1 > \gamma_2 > \ldots > \gamma$. At each stage $m$, starting with $N$ trajectories we take two steps. First, we discard trajectories that do not fall below threshold $\gamma_m$. Second, we replenish back up to $N$ trajectories. To replenish discarded trajectories, we clone a random un-discarded $\tau^{(1)}$ up to $t'$—the first time step where $\mathcal{L}(\tau^{(1)}, \varphi, 0) < \gamma_m$. Then, we re-simulate $\tau^{(1)}$ from $t'$ to $T$. This ensures all $N$ trajectories at stage $m$ are below $\gamma_m$. With staged, partial re-samplings, Eq (1) becomes:

$$\prod_{m=1}^{M} P(\mathcal{L}(\varphi, \tau, 0) < \gamma_m | \mathcal{L}(\varphi, \tau, 0) < \gamma_{m-1})$$ (11)

Given enough levels, each conditional probability should be significantly larger than $P(\mathcal{L}(\varphi, \tau, 0) < \gamma)$. The final computation:

$$\hat{p}_{ams} = \left\{ \prod_{m=1}^{M} \frac{N - K_m}{N} \right\} \times \frac{1}{N} \sum_{i=1}^{N} \mathcal{I}\{\mathcal{L}(\varphi, \tau^{(i)}, 0)\}$$ (12)
has $M$ stages and $N$ initial simulations, where $K_m$ is the number discards per stage. Unlike adaptive importance sampling, AMS guarantees convergence as $N \to \infty$ [8].

C. Adaptive Splits via Online STL Monitors

To achieve our high-level goal of adapting AMS to AV testing of STL specifications, we currently face a slight computational dilemma: Our robustness metric $L(\tau, \varphi, 0)$ defines a single batch robustness value from the start to the end of a complete trajectory, and takes computation time proportional to the length of $\tau$. However in Section (III-B), we saw that the discard and replenishment steps require access to the robustness values at arbitrary prefixes of trajectories. In other words, AMS re-simulation requires online computation of all partial trajectories:

$$\left\{ L(\tau^{(i)}_{[0:t']}, \varphi, 0) \mid t' \in [0, T], i \in [0, N] \right\}$$ (13)

We resolve this dilemma by taking a key insight from the online monitoring literature—we can interleave partial computations into the AMS process. To achieve this interleaving, we must first slightly alter our definition of $L$ from [9] to define our metric in terms of partial rather than full trajectories. Eq (14) is a modified metric $L_n$ (where $n$ references the “nominal semantics” of [13], augmented with past operators). This definition makes explicit that we only partially evaluate trajectory $\tau$ up to fixed time step $t$:

$$L_n(\rho(x) > 0, \tau_{[0:t]}, i) = \rho(\tau_{[i:]})$$

$$L_n(\neg \varphi, \tau_{[0:t]}, i) = -L_n(\varphi, \tau_{[0:t]}, i)$$

$$L_n(\Box_i \varphi, \tau_{[0:t]}, i) = \inf_{i' \in [i+1, t]} (L_n(\varphi, \tau_{[0:i]'}, i'))$$

$$L_n(\Diamond_i \varphi, \tau_{[0:t]}, i) = \sup_{i' \in [i+1, t]} (L_n(\varphi, \tau_{[0:i]'}, i'))$$

$$L_n(\bigvee_i \varphi, \tau_{[0:t]}, i) = \sup_{i' \in [i+1, t]} (L_n(\varphi, \tau_{[0:i]'}, i'))$$

$$L_n(\bigwedge_i \varphi, \tau_{[0:t]}, i) = \inf_{i' \in [i+1, t]} (L_n(\varphi, \tau_{[0:i]'}, i'))$$

$$L_n(\varphi U t \varphi_2, \tau_{[0:t]}, i) = \inf_{i, i_2 \in [i, i_2]} \left( L_n(\varphi_2, \tau_{[0:i_2]}, i_2), \min_{i_2} \left( L_n(\varphi_1, \tau_{[0:i_1]}, i_1) \right) \right)$$ (14)

With the above definition, we could now naively compute all members of (13) by re-evaluating $\varphi$ at every $i$, up to every $t$, at every re-sampling step. This is computationally wasteful, as the robustness values of many partial trajectories share many operations with the computations of their prefixes. To take advantage of this fact, we instead maintain a work-list for each $\tau$. A work-list stores a mapping from specification $\varphi$ and time step $t$ to robustness value $L_n(\varphi, \tau_{[0:t]}, 0)$. By using a work-list, we can obtain a robustness value for the partial trajectories at time $t+1$ using just the newly available state $x_{t+1}$ and the previous values of the work-list, rather than repeating computations over the entire trajectory.

Alg (1) takes as input the current work-list at time step $t$ and sketches how it is updated online using the newly arrived state $x_{t+1}$. For predicates $\rho(x)$, incoming state $x_{t+1}$ is added only if it is in $\varphi$’s time horizon. For example, for $\varphi = \Box_{[0.2]}(\rho(x) > 0)$, state $x_3$ would not be added, as it falls outside the relevant interval. For formulas like negation and conjunction, pointwise operators are leveraged to combine the existing results from previous sub-formula computations. Similarly for temporal operators, we can use the sliding min-max algorithm of [24] to compute running maxes over the relevant sub-formula intervals. Further optimizations can be added (e.g., replacing chunks of a work-list with ‘summaries’ as sufficient information arrives [13]) but we omit the details here. We can access the robustness of a partial trajectory from the updated work-list by querying $w\text{-list}[\varphi][0]$.

Algorithm 1: Update Work List (Adapted from [13])

1. Function $w\text{-list}(\varphi, x_{t+1})$
2. switch $\varphi$ do
3. case $\varphi > 0$ do
4. case $\varphi$ do
5. case $\varphi \land \varphi_2$ do
6. case $\varphi U t \varphi_2$ do
7. case $\varphi H t \varphi$ do

Now that we can compute and cache STL robustness values for partial trajectories, we can interleave this with our AMS sampler. Alg (2) provides an overview of our sampling technique for computing the probability that our algorithm will hit (or violate) its specification.

Lines (1-6) generate $N$ initial trajectories by simulating perceptual observations, control actions, and forward dynamics as outlined in sections (II-A-II-B). Lines (7-8) track the STL robustness value of trajectories at every intermediate time step by maintaining up-to-date work-lists as described in Alg (1). Lines (9, 17) adaptively set discard thresholds $\gamma_m$ such that the $K$ safest trajectories are discarded at each
stage. To replenish those discarded trajectories, lines (14-16)
copy one of the remaining un-discarded \( \tau^{(i)} \) up until the first step \( t \) where the robustness value of the partial trajectory falls below \( \gamma_m \). We then re-simulate starting from \( j \) to produce a new trajectory. We repeat this process until the discard threshold \( \gamma_m \) falls below the desired failure threshold \( \gamma \), then calculate a final estimate \( \hat{p}_{ams} \) using Eq (12).

IV. EXPERIMENTS

The following experiments use our running example of an AV lane-change maneuver to evaluate our method. They demonstrate that Alg (2) can provide failure estimates for a full black-box AV-system across multiple common traffic rules. When compared to baselines, we find Alg (2) provides more accurate failure estimates in fewer simulations. Further, we investigate how sampling performance differs across discard-rates and rule types. Fig (1) shows our CommonRoad simulation setup [2]: The ego starts in the centre-lane at 15 m with velocity 20 m/s. Its primary goal is to track a reference velocity of \( u_R=30 m/s \). Three obstacles impede it—a static obstacle at 40 m (forcing the ego to change lanes); A centre-lane vehicle at 50 m with velocity 5 m/s, which cuts into the overtake lane after 0.6 seconds (slowing the ego or forcing a lane change); a right-lane vehicle at 50 m, velocity 10 m/s, which merges after 1 second (preventing the ego from slowing abruptly). Simulations last \( T=40 \) steps (4 s, \( \Delta t=0.1 s \)). Dynamics evolve according to a kinematic single-track model [37].

Our AV-system under test uses a standard pre-trained lidar-based obstacle detector—OpenPCDet’s Multi-Head PointPillar [41]. As described in Section II-A, we train a surrogate PEM to replicate the behaviour of this detector in simulation.

The PEM is composed of two separate Gaussian Processes: The first is a binary classifier, which predicts whether the lidar perception system would have failed to detect a given obstacle. The second is a regression model, which predicts how much noise would typically be added to the true location of an obstacle’s bounding box. The GP’s were fitted using Pyro [4], with an RBF kernel and sparse variational regression with 100 inducing points. As training data for fitting the GPs, we used 65500 entries from the NuScenes Lidar Validation Set [3]. The GP input features were a 7-d vector—the x/y obstacle position, rotation, length/width/height dimensions, obstacle “visibility category” (where “1” means \( \leq 40\% \) occlusion; “4” means \( \geq 80\% \)). The GP outputs consist of a 1-d binary variable for successful/unsuccessful obstacle detection, and a 3-d real-valued variable for the offsets between the obstacle’s true x/y/rotation and the PointPillar estimate.

For tracking and predicting future vehicle locations, we used the interacting multiple models (IMM) filter for lane-changes from [7]. This method operates similarly to a typical kalman filter, but instead of making estimates based on a single model, it maintains estimates from multiple models (i.e., for whether the vehicle will stay in the current lane, switch to the left lane, or switch to the right lane) and merges those estimates based on each model’s current likelihood.

For model predictive control (Eq (7)) we use the lane-change controller from [26]. At a high level, its cost function \( J \) is comprised of 8 sub-goals: Reach a target destination; track a reference velocity, minimize acceleration, turn velocity, jerk and heading angle; stay close to the centre of the nearest lane; and avoid entering the “potential field” of other obstacles. To solve (7), we used the Gurobi [20] optimizer. To ensure a feasible control action is always available, we first pre-solve a convex simplification of (7) with CVXPy [14] (following [18]). For full implementation details of the cost-function sub-goals (and the weights used to balance them) see [26] and the associated code for our paper.

We test our sampling method with respect to 4 formalizations of rules from the Vienna Convention on Road Traffic (taken from [29]). Table (1) shows the STL formulas for each rule. Full definitions of individual predicates (in_same_lane, drives_faster etc.) are in [29], but high level descriptions of each rule are as follows: \( \varphi_1 \)—maintain a minimum distance from vehicles in front (proportional to vehicle speed). If a vehicle “cuts in” from an adjacent lane, the ego gets \( t_{cut} \) seconds to re-establish distance. \( \varphi_2 \)—never drop acceleration below “unnecessary” levels (relative to vehicles in front). \( \varphi_3 \)—velocity should never fall below some minimum level (unless stuck in traffic). \( \varphi_4 \)—do not exceed the speed of left-lane vehicles unless merging from an access lane, or left-lane traffic is slow moving.

Our algorithm (listed as STL-AMS below) uses \( N=250 \) starting simulations, a discard amount of \( K=25 \), and final failure threshold of \( \gamma = 0 \). We compare against three base-

**Algorithm 2: Online STL-AMS**

```
for \( i \in [1, N], t \in [0, T] \) do
  \( \epsilon_t, y_t, x_t, a_t \leftarrow \) Sample observations from PEM [56].
  Append \((x_t, a_t)\) to trajectory \( \tau^{(i)} \)
  // Maintain work-list per trajectory (Alg [41])
  \( w-lst_{t+1}(i) \leftarrow upd-wl(w-lst_t(i), \tau^{(i)}) \)
  \( L_{t+1}(i) \leftarrow w-lst_{t+1}(i)[\varphi(0)] \) // Robustness of \( \tau^{(i+1)} \)
  \( x_t+1 \leftarrow \) Step forward simulation
  Sort \((L_T(i), ..., L_T(N))\) then set \( \gamma_m \leftarrow L_T(K) \)
while \( \gamma_m > \gamma \) do
  \( m \leftarrow m + 1 \)
  Discard all trajectories \( \tau^{(i)} \) where \( L_T(i) \geq \gamma_K \)
  \( I_k \leftarrow \) Indices of remaining un-discarded trajectories
  for \( i \in [0, N] \setminus I_k \) do
    Select a random \( j \in I_k \)
    Find the first time step \( t' \) where \( L_T(i) < \gamma_k \)
    \( [\hat{x}_{t'}, [\hat{L}_{t'}(i)] } \leftarrow Copy values from \( \tau^{(j)} \)
    \( [\hat{x}_{t'}, [\hat{L}_{t'}(j)] } \leftarrow Re-simulate \( \tau^{(j)} \) from time \( t' \)
  Sort \((L_T(i), ..., L_T(N))\) then set \( \gamma_m \leftarrow L_T(K) \)
return \( p_{ams} \) via Eq (12)
```
lines: First, a Monte-Carlo sampler (Raw-MC), which runs \( N \) simulations, estimating via (10). Second, an importance sampler with a fixed proposal (Imp-Naive). We choose a proposal distribution which deliberately fails to detect 50\% of obstacles, and applies gaussian noise (\( \mu = 0, \sigma^2 = 1 \)) to the bounding boxes of those it does detect. Third, a neural network-based importance sampler with an adaptive proposal learned via the cross-entropy method (Imp-CE) from [21]. Its inputs and outputs are the same as those of the GP-PEM described previously. Similar to AMS, adaptive importance sampling proceeds in stages: At each stage \( m \) (for a total of \( M = 10 \) stages), \( N_m = 250 \) trajectories are sampled and sorted by robustness. Imp-CE then takes the lowest \( K = 0.1 \ast N_m \) trajectories\(^4\) and minimizes their KL-divergence under the target PEM versus current proposal. The intuition is that biasing our proposal towards the least robust trajectories in each stage should train a proposal which samples failure events with increasing probability.

### A. Results and Discussion

Table (II) shows estimated probabilities per rule. For “ground-truth”, we computed Eq (10) using 100000 simulations. Across rules, our method produces the most accurate estimates.

**Raw-MC** gives a reasonable estimate for \( \phi_1 \) (the least rare). For other rules with lower probability, raw sampling yields zero simulation failures, resulting in 0\% estimates. **Imp-Naive** produces non-zero estimates for all specifications except \( \phi_4 \), but vastly underestimates failure probability. This underscores the difficulty of fixed proposals— if a proposal distribution is too far from the original target, the likelihood weights per sample vary wildly, with estimates dominated by a tiny number of simulations. **Imp-CE** also produces unreliable estimates. Figs (3a)\[3c\] provide insights why: As learning progresses for \( \phi_3 \), the number of failures produced goes up, yet failure probability goes down. This suggests Imp-CE learns to bias its distribution towards a small set of unlikely failures, rather than approaching the true failure distribution. For \( \phi_4 \), robustness thresholds initially decrease, but flatline around \( m=6 \)—no failures found. We observe that this “flatlining” continues even past the \( M = 10 \) stages documented in this paper. This highlights how challenging it can be to learn relevant features given longer horizons and state-spaces.

The results of Table (II) are encouraging, but we found Alg (2) was sensitive to discard rate \( K \). Fig (4) shows how threshold levels evolve at each stage (for \( K \) values from 2 to 225). For \( \phi_4 \) (the rarest), we found that too low or high \( K \)'s caused unacceptable numbers of “extinctions”[9]—stages where all trajectories have identical robustness, rendering replenishment impossible.

Experiments demonstrate Alg (2) is viable for accurately estimating specification failure for a black-box AV-system. However, this case study looks only at a single interstate traffic scenario, and our experiments necessarily have limitations: We considered perceptual disturbance as the sole source of simulation stochasticity; vehicle starting configurations remained fixed. Such experiments could be extended by placing a prior distribution over starts [32], without altering the method. To trust our estimates, we also

<table>
<thead>
<tr>
<th>Rule Description</th>
<th>STL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safe Dist from Vehicles</td>
<td>( \square_{[0, \infty)} (\text{in_same_lane}(x_0^n, x^n) \land \text{in_front_of}(x_0^n, x^n)) \land \neg \Box_{[0, \infty)} (\text{cut_in}(x_0^n, x^n) \land \text{in_front_of}(x_0^n, x^n)) \land \text{keeps_safe_distance_prec}(x_0^n, x^n) )</td>
</tr>
<tr>
<td>Unnecessary Braking</td>
<td>( \square_{[0, \infty)} (\neg \text{unnecessary_braking}(x_0^n, x^{z_1 \ldots z_3})) )</td>
</tr>
<tr>
<td>Preserve Traffic Flow</td>
<td>( \square_{[0, \infty)} (\neg \text{slow_leading_vehicle}(x_0^n, x^{z_1 \ldots z_3}) \implies \text{preserves_flow}(x_0^n)) )</td>
</tr>
<tr>
<td>Don’t Drive Faster than Left Traffic</td>
<td>( \square_{[0, \infty)} (\text{left_of}(x_0^n, x_0^n) \land \text{drives_faster}(x_0^n, x_0^n)) \implies (\text{in_slow_traffic}(x_0^n, x^{z_1 \ldots z_3}) \land \text{slightly_higher_speed}(x_0^n, x_0^n)) \lor (\text{on_access_ramp}(x_0^n) \land \text{on_main_carriageway}(x_0^n)) )</td>
</tr>
</tbody>
</table>

---

\(^4\)Standard practice sets \( K \) in range \([0.01, 0.2] \ast N_m \) [11].
TABLE II: Estimated Failure Probabilities (5 Repetitions)

<table>
<thead>
<tr>
<th>Method</th>
<th>$\varphi_1$</th>
<th>$\varphi_2$</th>
<th>$\varphi_3$</th>
<th>$\varphi_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw-MC$_{250}$</td>
<td>1.2e-02 (±4.0e-03)</td>
<td>0.0 (±0.0)</td>
<td>0.0 (±0.0)</td>
<td>0.0 (±0.0)</td>
</tr>
<tr>
<td>Imp-naive</td>
<td>1.4e-08 (±2.5e-08)</td>
<td>2.8e-08 (±8.4e-08)</td>
<td>2.0e-08 (±3.9e-08)</td>
<td>0.0 (±0.0)</td>
</tr>
<tr>
<td>Imp-CF</td>
<td>7.1e-06 (±2.1e-05)</td>
<td>7.6e-22 (±2.2e-21)</td>
<td>4.4e-18 (±8.9e-18)</td>
<td>0.0 (±0.0)</td>
</tr>
<tr>
<td>STL-AMS (ours)</td>
<td>8.8e-03 (±6.2e-03)</td>
<td>1.5e-03 (±2.1e-03)</td>
<td>4.7e-03 (±3.4e-03)</td>
<td>3.5e-04 (±1.8e-03)</td>
</tr>
<tr>
<td>Ground Truth</td>
<td>9.1e-03</td>
<td>2.0e-03</td>
<td>3.6e-03</td>
<td>4.8e-05</td>
</tr>
</tbody>
</table>

![Fig. 4: STL-AMS robustness thresholds by stage.](image)

Assume our simulator accurately represents reality. Whilst outside this paper’s scope, clearly this assumption may not hold: Our PEM may be an inaccurate surrogate of the test domain (i.e., it would be beneficial to incorporate work on ML-uncertainty calibration [19]). Our scenario also had fixed traffic behaviour, but real traffic is reactive and stochastic. Other works explore these issues in detail [25]. Finally, while it can be seen as an advantage that our method adaptively selects an appropriate number of simulations, and re-uses results from previous simulations, these advantages complicate comparisons of our method to others in terms of sample efficiency. In future work, we aim to compare performance across a wider range of scenarios in terms of fixed computational effort across the full sampling pipeline.

V. RELATED AND FUTURE WORK

This paper estimates failure probabilities. Similar tasks include falsification (find one failure) and adaptive stress testing (find the most-likely failure) [11]. Such tasks do not directly accomplish our goal, but may contain insights for rapidly guiding initial simulations towards failure areas. A related task is synthesis—constructing controllers which explicitly obey $\varphi$ [1]. While synthesis can enforce adherence to specifications expressed in tractable STL subsets, the perceptual and control uncertainty in AV scenarios means testing remains necessary.

Combining splitting and logic has been attempted previously [22]. Rather than use STL robustness, such works restrict themselves to heuristic decompositions of linear temporal logic formulae. This renders them unsuitable for cyber-physical domains like AV.

One of our experiment baselines was importance sampling. Proposals are often represented by exponential distributions, since those have analytic solutions [45]. Neural networks have also been used to represent the proposal (as we did), [31]. Most work considers rarity in the context of vehicle configurations or behaviour. Our work instead considers rarity in the context of perceptual disturbances. A sampler category unexplored in this paper are markov-chain methods [5]. With appropriate assumptions on metric smoothness and system linearizability, such techniques have shown promise in domains with long chains of dependent states [40]. While out of scope, integrating such techniques with online STL monitoring may prove fruitful.

Despite the asymptotic normality of AMS, both splitting and sampling lack guarantees on estimation error for fixed $N$. Certifiable sampling addresses this with efficiency certificates [3]—customized samplers with a bound on sampling error relative to $N$. However, certification methods depend on augmenting existing failure estimation algorithms, so techniques from this paper remain relevant.

Online and offline algorithms exist to calculate STL robustness [16], [13]. Typically, their efficiency is not considered in the context of a sampling regime. Yet recent advances in online monitoring could be leveraged within AMS to discard infeasible trajectories early. For example, incorporating system dynamics, or causality [44], [12].

Our experiments target Interstate lane changes. Others encode rules for intersections, and situational awareness [15], [28]. In future work, we aim to assess sampling effectiveness across this diversity of specifications.

REFERENCES

benchmarks for motion planning on roads. In: 2017 IEEE Intelligent
[3] Arief, M., Bai, Y., Ding, W., He, S., Huang, Z., Lam, H., Zhao,
D.: Certifiable deep importance sampling for rare-event simulation of
N., Karaletsos, T., Singh, R., Szerlip, P.A., Horsfall, P., Goodman,
N., Karaletsos, T., Singh, R., Szerlip, P.A., Horsfall, P., Goodman,