Risk-Aware Scene Sampling for Dynamic Assurance of Autonomous Systems

Shreyas Ramakrishna, Baiting Luo, Yogesh Barve, Gabor Karsai, and Abhishek Dubey
Institute for Software Integrated Systems, Vanderbilt University

***Abstract*—Autonomous Cyber-Physical Systems must often operate under uncertainties like sensor degradation and shifts in the operating conditions, which increases its operational risk. Dynamic Assurance of these systems requires designing runtime safety components like Out-of-Distribution detectors and risk estimators, which require labeled data from different operating modes of the system that belong to scenes with adverse operating conditions, sensors, and actuator faults. Collecting real-world data of these scenes can be expensive and sometimes not feasible. So, scenario description languages with samplers like random and grid search are available to generate synthetic data from simulators, replicating these real-world scenes. However, we point out three limitations in using these conventional samplers. First, they are passive samplers, which do not use the feedback of previous results in the sampling process. Second, the variables to be sampled may have constraints that are often not included. Third, they do not balance the tradeoff between exploration and exploitation, which we hypothesize is necessary for better search space coverage. We present a scene generation approach with two samplers called Random Neighborhood Search (RNS) and Guided Bayesian Optimization (GBO), which extend the conventional random search and Bayesian Optimization search to include the limitations. Also, to facilitate the samplers, we use a risk-based metric that evaluates how risky the scene was for the system. We demonstrate our approach using an Autonomous Vehicle example in CARLA simulation. To evaluate our samplers, we compared them against the baselines of random search, grid search, and Halton sequence search. Our samplers of RNS and GBO sampled a higher percentage of high-risk scenes of 83% and 92%, compared to 56% 66% and 71% of the grid, random and Halton samplers, respectively.**

***Index Terms*—Cyber-Physical Systems, Dynamic Assurance, Scenario Description Language, Bow-Tie Diagram**

I. Introduction

The widespread use of autonomous Cyber Physical System (CPS)[1] has often required them to operate under uncertainties like sensor degradation and shifts in the operating conditions, which increase its operational risk. Design-time Assurance Case [1] with risk assessment information is used to argue the system's safety at runtime. However, the dynamically changing operating conditions of the system at runtime potentially invalidate the design-time assumptions and the safety arguments [2]. So, a dynamic assurance approach with proactive safety assessment components like Out-of-Distribution (OOD) detectors [3] and dynamic assurance monitors [4] is required for runtime safety assurance. Designing these components often requires labeled data from different operating modes of the system that belong to scenes with adverse operating conditions and sensor or actuator faults. These scenes are referred to as risky scenarios [5] or safety-critical scenarios [6]; in this paper, we refer to them as high-risk scenes.

Often the data related to high-risk scenes are under-represented in the training sets [7], leading to a data imbalance problem. If we can generate these under-represented events, they can be used to design the safety assessment components required for dynamic assurance and retrain the controller LECs to improve their accuracy [8]. However, collecting real-world data of such high-risk scenes can be expensive and slow in real-world conditions. Synthetic data from simulators have been used to address this problem in engineering design and testing applications. Samplers are used to generate data across the search space created by the system parameters. For example, tools like Dakota [9] provide efficient samplers like incremental sampling, importance sampling, and adaptive sampling for uncertainty quantification in engineering design. Recently, this concept of sampling-based data generation is being adapted for autonomous systems [5], [10]–[12]. Domain-specific Scenario Description Language (SDL) like Scenic [8], and MSDL [13] with conventional samplers like random and grid search are integrated with simulators like CARLA [14] to generate high-risk scenes.

Despite their success in generating high-risk scenes, we point to several limitations in using the conventional samplers. First, they perform *passive sampling*, which does not use the feedback of previous results in the sampling process. Second, the scene variables (e.g., environmental conditions) being sampled typically have *sampling constraints* and *co-relations* that need to be considered. For example, environmental conditions (e.g., precipitation) may have physical constraints on their values that govern their temporal evolution. Applying these constraints is necessary for generating meaningful scenes [8]. However, the conventional samplers do not include these sampling constraints. Third, conventional samplers do not balance the *exploration vs. exploitation trade-off*. For example, random and Halton searches prioritize uniform search space coverage, so they only explore. In contrast, grid search aims to cover a given grid exhaustively, so they only exploit it. However, as discussed by Jerebic, Jernej *et al.* [15], balancing these strategies can result in higher coverage diversity which is commonly measured using clustering properties like the number of clusters and cluster population.

To address these limitations, we present a scene generation approach that has a domain-specific SDL integrated with two sampling approaches for generating high-risk simulation scenes. The key contribution of this work is two sampling approaches called Random Neighborhood Search (RNS) and

[1]CPS with Learning Enabled Component (LEC)

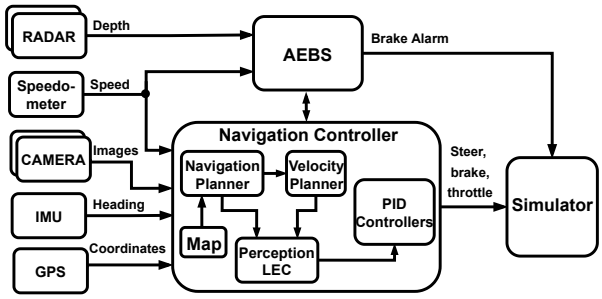

Fig. 1: AV system model designed for the CARLA autonomous challenge [17] setup. The AV is primarily driven by an LEC-based navigation controller adapted from Chen et al. [18]. We have augmented an Automatic Emergency Braking System (AEBS) supervisor for emergency braking.

Guided Bayesian Optimization (GBO), which perform active sampling by using previous simulation results in the sampling process. They also include sampling constraints that govern the evolution of scene variables. In addition, they provide explicit hyperparameters to control the tradeoff between exploration vs. exploitation of the search strategy. Further, to facilitate the samplers, we use a novel risk-based scoring function that evaluates how risky the scene was for the system. We demonstrate our approach using an Autonomous Vehicle (AV) case study in the CARLA simulator [14]. To evaluate our samplers, we compared them against the baselines of random search, grid search, and Halton sequence search [16] in terms of three metrics that measure the total high-risk scenes sampled, the sample diversity, and the search times. Our experimental evaluation shows that the RNS and GBO outperform the baselines in terms of the total high-risk scenes and diversity metrics. Also, the RNS sampler had comparable search times to the baselines, but the GBO sampler has a higher search time. The source code is available online[2].

The outline of this paper is as follows. In Section II, we introduce the background needed to understand this work. In Section III, we describe the problem statement, followed by a description of our approach in Section IV. In Section V, we implement and evaluate the samplers in the context of an AV case study in the CARLA autonomous driving challenge [17]. This setup requires an AV (See Fig. 1) to navigate an urban town setting with complex traffic scenarios, adverse weather conditions, and sensor faults. Finally, we present related research in Section VI followed by conclusions in Section VII.

II. Background

A. System Design Procedures

The typical design procedure of autonomous CPSs include the design, training, testing, and deployment phases, which we categorize into five steps for designing our systems: (1) *Design Phase*, which involves system analysis, hazard analysis, and Assurance Case construction. (2) *Training Phase*, which involves collecting training scenes and training the LECs on these scenes. (3) *Calibration Phase*, which involves calibrating the detectors [3], [19], and dynamic assurance monitors (Section II-B). Calibration requires curating a *calibration set* that includes scenes with sensor faults and adverse weather in addition to the training scenes. To generate these scenes, we use both random and grid samplers. We use the random sampler to find the conditions affecting the system and then use the grid sampler to generate more scenes around it. (4) *Testing Phase*, which involves generating high-risk scenes for testing the trained system. (5) *Deployment Phase*, which involves deploying the trained and tested system to operate.

In particular, the design phase is important because it consists of gathering and documenting information about the system's goals, requirements, operating conditions, and component faults. It also includes designing the system models like the architecture model (e.g., Fig. 1) and the system function breakdown model. Following this, the designers perform an *hazard analysis*, which involves identifying the hazards to the system that will result in a system consequence. The system could have a collection of hazards $\{H_1, H_2, \cdots, H_n\}$ stemming from its operation, functioning, software components, and hardware components. For example, in the context of an AV, operational hazard elements can be the movement of traffic participants including pedestrians, and other vehicles. Any of these hazards in a given condition can result in a system consequence. This enables creation of Bow-Tie Diagram (BTD) [20], a model describing the chain of events $t_i \rightarrow b_p \rightarrow e_{top} \rightarrow b_m \rightarrow c_i$, where t_i are system threats, b_p are preventive barriers, e_{top} is a top event, b_m is a mitigation barrier, and c_i is a system consequence. Fig. 2 describes the operational hazard of "roadway obstruction" for the AV case study. The BTD has two system threats: vehicles (T1) and pedestrians (T2) in the path of the ego vehicle. These threats can escalate to become a top event (TOP) if not prevented by the preventive barriers $B1$ and $B2$. Next, the TOP event can escalate to become a system consequence if not mitigated by the barrier $B3$. During the design, these hazard analyses are used to create assurance cases [1], showing that the top-level goals of the system are satisfied. Further, the BTD allows for dynamically monitoring the likelihood of system consequences through a technique called *dynamic assurance*.

B. ReSonAte

To perform dynamic assurance, we leverage our previously developed tool called Runtime Safety Evaluation in Autonomous Systems (ReSonAte) [4], which uses an augmented BTD derived from the hazard analysis. The augmentation adds various event probabilities for each causal chain, conditioned on the state of the system and the environment, including sensor failures, actuator failures, the output of anomaly detectors, and environmental conditions. Specifically, we need to estimate the (a) the conditional probability of the barrier's success in each state $f_b(x = (B_i, s))$, and (b) the frequency of occurrence of a threat $f_e(T_i, s)$ in each state. We infer these conditional relationships from the calibration dataset gathered during the previously discussed calibration phase. The estimated probabilities are stored in Lookup Tables (LUTs) and used at runtime to calculate the hazard rate λ and the

[2]https://github.com/scope-lab-vu/Risk-Aware-Scene-Generation-CPS

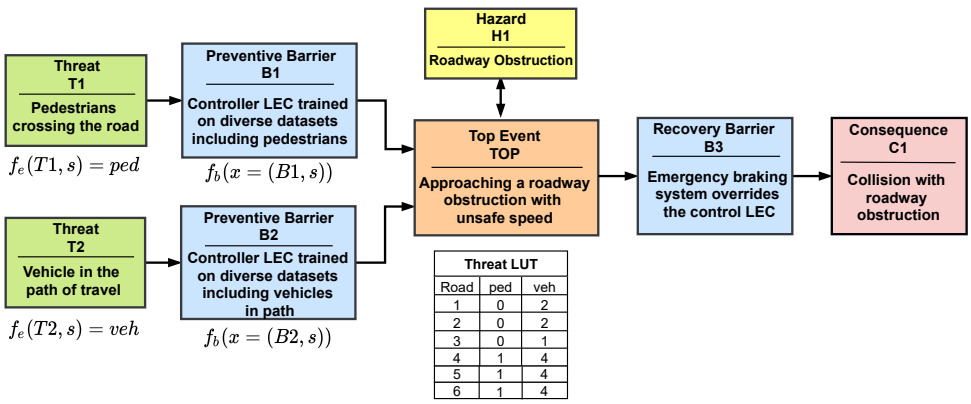

Fig. 2: Bow-Tie Diagram for the AV case study adopted from our previous work [4]. Each block includes information about the event type and its description. The conditional probability and the threat LUTs are estimated from calibration data, which is discussed in Section V-A4.

likelihood of the hazard occurrence in each time unit (t) as $1 - e^{-\lambda \cdot t}$.

III. Problem Formulation

We consider the autonomous CPS to be operating in an environment, characterized by two sets of variables $\mathbb{E} : x \rightarrow \Re^+$ and $\mathbb{S} : y \rightarrow \Re \times \Re$. $\mathbb{E}$ are the environmental variables like rain, traffic density, time-of-the-day. $\mathbb{S}$ are structural variables related to roadway features and are characterized by waypoints w denoted by a two-dimensional matrix of latitude and longitude. Also, we can map each waypoint to a road segment. In addition, we consider the set of faults in the system sensors and actuators $F : x \rightarrow 0, 1$. For a particular operating environment, there can be several environment variables $e \in \mathbb{E}$, several possible waypoints $w \in \mathbb{S}$, and several sensors in the system that can fail $f \in \mathbb{F}$. The CPS is trained through a collection of scenes, where each scene s_i is an ordered sequence of k sampled observations $< \mathbb{E}_i, \mathbb{S}_i, \mathbb{F}_i >_{i=1}^{k}$. We collectively refer to these variables as the scene variables s_v. Given each scene, we can associate the hazard rate λ for every hazard identified for the system. The system can have a collection of hazards $\{H_1, H_2, \cdots, H_n\}$ stemming from its components (e.g., software, hardware) or its operating conditions. Each identified hazard will have an associated hazard rate estimated using the dynamic assurance routines ($DA : s_i \rightarrow \lambda$) introduced in Section II-B. We can use the estimated hazard rate to compute the likelihood of the hazard occurrence ($1-e^{-\lambda \cdot t}$) in each time unit t. The likelihood of hazard for different hazard conditions put together constitutes the system's operational risk S_{Risk}.

Also, given a sequence of samples in the scenes, physical constraints govern the temporal evolution of the sampled values. For example, a constraint on an environmental variable $e \in \mathbb{E}$ governs that $|s_i^e - s_{i+1}^e| \leq \alpha_e \in \Re^+$. Similarly, the waypoints across a scene are governed by routability property. To explain this, we use the sampling constraints $\mathcal{SC}$ in Table I, which we applied for an AV case study, discussed in Section V-A. The constraint governs the road segment such that the distance between any two selected waypoints cannot exceed 10 meters. This constraint regulates that the road segment is orderly selected. The environmental variables have constraints on how their values change between consecutive scenes. The traffic density has constraints on its value and is also dependent on the type of road segment (e.g., crossroads, intersections). Also, we do not place any constraints on the fault variables. Now, given this setup, we want to address the following problem.

Variable Type	Name	Range	Constraints
Structural	Road Segments (RS)	[0,9]	$wp_1 - wp_2 \leq 10m$
Environmental	Precipitation (P)	[0,100]	$P_i - P_{i+1} \leq 5\%$
	Time-of-Day (T)	[0,90]	$T_i - T_{i+1} \leq 10^\circ$
	Cloud (C)	[0,100]	$C_i - C_{i+1} \leq 5\%$
	Traffic Density (TD)	[0,20]	TD_i depends on RS_i $TD_i - TD_{i+1} \leq 10$
Faults	Cam Blur	[0,1]	N/A
	Cam Occlusion	[0,1]	

TABLE I: Scene Variables, their distribution ranges and sampling constraints for the CARLA AV case study introduced later in Section V.

Problem. *Given a threshold δ (computed across the calibration set of scenes), sample N scenes that maximizes the risk* $\max(sum(S_{Risk} - \delta))$. *In addition we want to ensure that the scene variables are diverse. We call a scene with $(S_{Risk} - \delta) > 0$ as a high risk scene.*

While the first objective measures the sampler's capability to sample high-risk scenes, the second measures its coverage diversity, representing its capability to balance exploration vs. exploitation of the search space. A sampler with high diversity is desirable for better coverage of the search space [15]. We measure diversity in terms of a quantitative metric (measured using the number of the optimal clusters, as measured using silhouette score [21]) and a qualitative metric (measured using the variance of risk across each scene across all clusters).

IV. Proposed Approach

We now describe the approach we use to generate high-risk scenes for AV in simulation. Overview of the approach is illustrated in Fig. 3 and described in detail below.

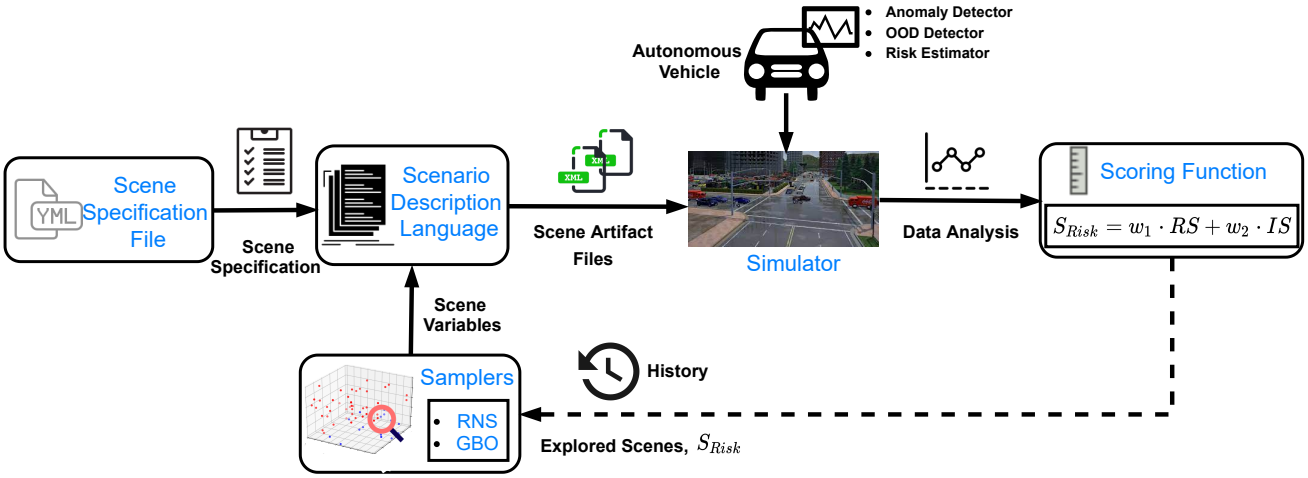

Fig. 3: Overview of our approach for generating high-risk scenes. The samplers perform active sampling using the feedback loop represented by the dotted line.

```
scene sample {
type int
type string
class scene_info {
    town: town_description
    weather: weather_description
    road: road_segments
    traffic: traffic_density
    faults: sensor_faults }
entity town_description{
        id:string
        map:string  }
entity weather_description {
    precipitation: distribution
    cloudiness: distribution
    time-of-day: distribution }
entity distribution {
        low: int
        high: int  }
}
```

Fig. 4: This listing shows a fragment of the scene description for CARLA simulation that was generated using our SDL.

A. Scene Generation

We have designed a scene generation approach, which includes a SDL integrated with samplers to sample scenes over different scene variables introduced in Section III. The SDL is designed using the textX [22] meta-language and it has two components. (a) *grammar*, which consists of a set of rules required to define a scene in the meta-language. (2) *meta-model*, which defines the concept of a scene using scene variables and their distribution ranges. In this setup, a scene $s = \{e_1,e_2,...,e_k\}$ is a collection of entities, where each entity is a tuple $< s_v, v_d, v_c >$ of the scene variables s_v, distribution properties v_d (e.g., distribution type, ranges) and variable constraints v_c. The SDL also has an interpreter to generate the scene artifact files that run the simulator. The interpreter also abstracts the complexity of the SDL from the user by providing a scene specification file with scene variables and samplers that can be chosen to generate different scenes. A fragment of the scene description for CARLA simulation is shown in Fig. 4.

B. Scoring Function

We have designed the *Risk Score* metric (S_{Risk}) to evaluate the sampled scene. This score is a weighted combination of two utility functions and is computed as $S_{Risk} = w_1 \cdot RS + w_2 \cdot IS$. The utility functions are. (1) *ReSonAte Score* (RS), which measures the system's risk in an operational scene using the hazard rates calculated by the ReSonAte framework. (2) *Infraction Score* (IS), which measures the actual infractions performed by the system. We use equal weights of $w_1 = w_2 = 1$ for both these utility functions to ensure that the estimated risk and the actual risk caused by infractions are weighted equally in the subsequently sampled scenes. In the future, we will formulate this metric as a multi-objective optimization problem as performed by Viswanadha, Kesav, *et al.* [12].

1) ReSonAte Score (RS): Measures the average risk of the system's failure across a scene. We leverage the ReSonAte framework which estimates the hazard rate λ, which is averaged across the scene using $\frac{\int_{T_1}^{T_2} \lambda \, dt}{T_2 - T_1}$ to compute the system's risk or the ReSonAte score. Where λ is the estimated hazard rate and T_1, T_2 are the start and end time of the scene.

2) Infraction Score (IS): Measures the system's actual infractions across a scene. This score can include a variety of infractions like route deviation, lane violation, and traffic rule violation. These infractions (I_k) can also be weighted (w_k) according to their severity and later summed together to compute a unified infraction score $IS = \sum_{k=1}^{n} w_k \cdot I_k$. To illustrate, the infraction score for the AV case study was computed as $IS = 0.7 \cdot I_S + 0.8 \cdot I_R + I_{RD}$. Where I_S is running a stop sign infraction, I_R is running a red-light infraction, and I_{RD} is the deviation in the route taken by the AV. We adopted these weights from the CARLA challenge setup [17].

Assuming these scores are available at the end of each scene, we compute the risk score S_{Risk}. Further, a scene is classified to be of high-risk if $S_{Risk} - \delta > 0$. Where δ is the risk threshold computed across the previously encountered calibration scenes. Considering these scenes are sampled from

Algorithm 1 Random Neighborhood Search

Parameter: number of iterations t, explored list $\mathcal{E}$, neighborhood size k
Input: search space $\mathcal{D}$, sampling constraints $\mathcal{SC}$, scene variables s_v, threshold δ
Output: List of scenes

for $x = 1, 2, ..., t$ **do**
  if $S_{Risk} < \delta$ or ($S_{Risk} > \delta$ and $N \geq k$) **then**
    Randomly sample s_v within $\mathcal{D}$ to generate random scene R
  else
    Apply $\mathcal{SC}$ to create bounded search area $\mathcal{B}$
    Randomly sample s_v within $\mathcal{B}$ to generate neighboring scene N
    Apply kd-tree to find if there are atleast k neighbors (N) in $\mathcal{E}$
  end if
  Use the sampled variables to generate the scene artifact files
  Simulate the scene and compute the S_{Risk}
  Append sampled variables and S_{Risk} to $\mathcal{E}$
end for

Algorithm 2 Guided Bayesian Optimization

Parameter: number of iterations t, initial iterations k, explored list $\mathcal{E}$
Input: search space $\mathcal{D}$, sampling constraints $\mathcal{SC}$, scene variables s_v, threshold δ
Output: List of scenes

for $x = 1, 2, ..., t$ **do**
  if $x \leq k$ **then**
    initialize GP model with random samples s_v from $\mathcal{D}$
  else
    Apply $\mathcal{SC}$ to create bounded search area $\mathcal{B}$
    Use μ_t and σ_t in the UCB function to sample s_v within $\mathcal{B}$
  end if
  Use the sampled variables to generate the scene artifact files
  Simulate the scene and compute the S_{Risk}
  Append sampled variables and S_{Risk} to $\mathcal{E}$
  Update the GP model using $\mathcal{E}$. Update μ_t and σ_t
end for

the same underlying distribution, we select the threshold at the 95^{th} percentile of S_{Risk} for every scene in the calibration set.

C. Samplers

We have developed two samplers called the Random Neighborhood Search and Guided Bayesian Optimization, which are extensions of the conventional random and Bayesian Optimization (BO) search. The extensions are. (1) *active sampling*, we use a feedback loop of previous results to sample the next scene variables. (2) *constraints-based search*, we create sampling constraints $\mathcal{SC}$ to add constraints on the scene variables. Table I shows the constraints that we applied for an AV case study. (3) *exploration vs. exploitation trade-off*, we introduce explicit hyper-parameters to balance the strategies.

1) Random Neighborhood Sampler (RNS): This sampler extends the conventional random search by including the kd-tree nearest neighborhood search algorithm [23]. We aim to add the missing exploitation capability to random search with this extension. Briefly, the approach works as follows. It initially explores a scene through random sampling and then exploits the area around the explored scene using the sampling constraints and the nearest neighbor search. Algorithm 1 illustrates the steps involved, and it works as follows.

First, the algorithm explores the search space $\mathcal{D}$ by randomly sampling the scene variables s_v from their respective distribution range v_d. These randomly selected variables are used to generate the scene artifact files, run the simulator and compute the risk score S_{Risk} for the scene.

Second, if the $S_{Risk} < \delta$, the statistics of the scene are stored, and the scene variables are re-sampled again from their respective distribution range v_d. However, if the $S_{Risk} > \delta$, the neighborhood around the randomly sampled scene R is exploited to generate more scenes. For this, we create a bounded region $\mathcal{B}$ around R by applying the sampling constraints $\mathcal{SC}$ to the distribution ranges of the scene variables.

Third, the algorithm uses the kd-tree nearest neighborhood search to find if there are at least k scenes generated in the neighborhood of the explored scene R. That is, it checks if R has at least k similar scenes in $\mathcal{E}$. To measure the similarity, we use the l_2 distance metric ($d(x, R) = ||x - R||_2$) and identify the similar scenes as $N = \{\forall s_l \in \mathcal{E} | d(R, s_l) < \tau\}$. Where R is the currently sampled scene, and s_l are previously explored scenes in the explored list $\mathcal{E}$. This step returns a list of neighbors whose distances to R are smaller than a threshold τ. If the number of the neighbors $N \geq k$, the search around R is stopped. Then, the first step is repeated to randomly explore a new scene in the entire search space $\mathcal{D}$. However, if $N \leq k$, then the search around R is continued. In this sampler, k is the hyperparameter used to control the exploitation.

Although the sampler can both explore and exploit, the exploration is performed uninformed without strategic knowledge about the search space. This could result in high-risk regions of the search space left unexplored.

2) Guided Bayesian Optimization (GBO): The uninformed exploration of the RNS sampler is addressed by our second sampler, which has two components to perform informed exploration and active sampling. First, a probability function model is fitted across all the previously explored variables. This feedback allows the model to learn the search space region with large uncertainty that will return a higher objective value. A Gaussian Process (GP) model is most used to fit the function. Second, an acquisition function that strategically finds the succeeding variables to optimize the objective function. We use the Upper Confidence Bound (UCB) [24] acquisition function that provides the β hyperparameter to control the exploration vs. exploitation trade-off (See Eq. (1)). We also include the sampling constraints to restrict the region where the acquisition function looks for the next sampling variables. Algorithm 2 shows the operation of the GBO sampler, which is discussed below.

First, the GP model with properties $\mu[0]$ and $\sigma[0]$ is initialized for first k iterations. During these iterations, the scene variables s_v are randomly sampled from their respective distribution range v_d. These randomly selected variables are used to generate the scene artifact files, run the simulator and compute the risk score S_{Risk} that is added to the exploration list $\mathcal{E}$ along with the sampled variables. After the k initial iterations, the initialized GP model is fitted across all the entries in $\mathcal{E}$ to calculate new posterior distribution $f(x_n)$ with updated $\mu[x_n]$ and $\sigma[x_n]$.

Second, the algorithm uses the sampling constraints $\mathcal{SC}$ to create a smaller search space $\mathcal{B}$ in which the acquisition

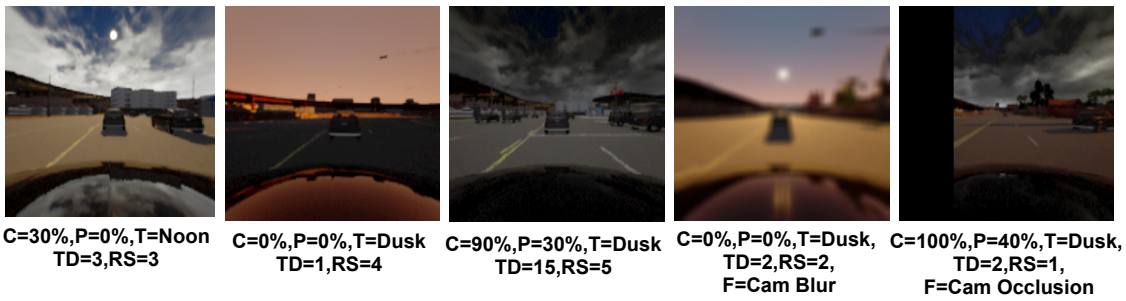

Fig. 5: Screenshots of different scenes generated for the CARLA AV case study. The description of these scenes are provided below the images.

function will sample the succeeding variables. The updated posterior distribution and the bounded search space created by the sampling constraints are used by an acquisition function to strategically select the following scene variables that will optimize S_{Risk}. In this work, we use the UCB [24] as the acquisition function, which is given below.

$$x_{n+1} = \underset{x \in \mathcal{D}}{\operatorname{argmax}}\, \mu[x_n] + \beta^{1/2} \cdot \sigma[x_n] \tag{1}$$

Where x_{n+1} are the newly selected variables that have the largest UCB. $\mu[x_n]$ and $\sigma[x_n]$ are the properties of the updated GP model. β is a parameter that allows for exploitation and exploration tradeoff. A larger value of β allows for higher exploration, and a lower value allows for exploitation. So, β needs to be carefully selected for optimizing the two strategies. The newly selected variables are used to run the simulation and compute S_{Risk}, which is added to the list $\mathcal{E}$ along with the selected variables. The steps of the algorithm are repeated for a specified number of iterations t.

The sampler has two problems that arise because of using the GP model. (1) Cold-start, which requires the model to be trained from scratch each time the sampler is used. This increases the sampling time. To address this, we use the knowledge of randomly sampled scenes from the previous runs to "warm start" the search process. (2) Scalability, the GP model suffers from a cubic time complexity [25], which limits its applicability to a large number of executions.

V. Evaluation

A. AV Case Study

Our testbed is an AV in the CARLA autonomous driving challenge [17] setup, which is needed to navigate an urban town with adverse weather conditions and sensor faults while avoiding collisions with vehicles and pedestrians in its travel path. The experiments with the simulator were performed on a desktop computer with AMD Ryzen Threadripper 16-Core Processor, 4 NVIDIA Titan XP GPUs, and 128 GiB RAM. The details of AV setup are discussed below.

1) System Model: The system block diagram of the AV is shown in Fig. 1. It uses a total of 9 sensors including, three forward-looking cameras, two radars, an Inertial Measurement Unit (IMU), a Global Positioning System (GPS), and a speedometer. It has a LEC based navigation controller, which is adapted from Chen et al. [18]. This controller uses a navigation planner that takes the waypoint information from the scene artifact file generated by our scene generation approach and divides the distance between the waypoints into smaller position targets. Then, it uses the GPS and IMU sensors to get the vehicle's current position and the next position to reach. Next, the position information is fed into a velocity planner to compute the vehicle's desired speed. The desired speed and camera images are fed into a perception LEC, which predicts the throttle and steering angle errors. These signals are sent to PID controllers to compute the throttle, brake, and steer control signals. Besides the LEC, there is also an AEBS, which uses the radar estimated object distance (r_d) with the vehicle's current speed to compute a "safe braking distance" (b_d). If $r_d \leq b_d$, a brake alarm is issued. The alarm overrides the throttle signals from the LEC. Finally, these control signals are sent to the simulator.

2) System Operational Phases: The AV is designed and operated in the following phases. In the *Design phase*, we design the system model and perform a hazard analysis to identify the operational hazards to the AV. Next, we identify sensor faults (camera blur and occlusion) that affect the AV's performance. In the *Training phase* we use an autopilot controller to collect data for training the perception LEC using the procedure discussed in [18]. In the *Calibration phase*, we train the detectors and the ReSonAte risk estimator. We train the detectors on the training set and the risk estimator on a calibration set, including scenes from the training set and additional scenes with sensor faults and adverse weather conditions, collected using random and grid samplers. Finally, in the *Testing Phase*, the trained LEC and the safety components are tasked to operate in 250 scenes generated by our samplers. These scenes included varying the weather conditions (cloud (C), precipitation (P)), time-of-day (T), traffic density (TD), road segments (RS), and sensor faults (F). The distribution ranges of these scene variables and their sampling constraints are listed in Table I. Fig. 5 illustrates the screenshots of a few CARLA scenes generated by our samplers.

3) Detectors: To detect camera-related faults such as blurred images and occluded images, each of the three cameras is equipped with OpenCV based blur detector and occlusion detector. The blur detector uses the variance of the Laplacian operator [26] to measure the blur level in the images. A high variance shows that the image is not blurred, and a low

variance (< 50) shows the image is blurred. The occlusion detector is designed to identify large blobs of continuous black image pixels. In our setup, if an image has connected black pixels $> 10\%$, then the image is said to be occluded; otherwise, it is not. The blur and occlusion detectors had an F1-score of 97% and 98% on the calibration set.

In addition, we designed a reconstruction-based β-VAE OOD detector [3] to identify shifts in the operating scenes. The detector model has four convolutional layers $16/32/64/128$ with (3x3) filters and (2x2) max-pooling followed by four fully connected layers with 2048, 1000, and 250 neurons. It also has a symmetric deconvolutional decoder structure and hyperparameters of β=1.2 and latent space=100. We trained the detector for 150 epochs using the 6000 camera images curated for training the AV. We tested the detector on a test set that included images from the training set and several images with high scene brightness, adverse conditions, and camera faults. For these images, the detector had an F1-score of 95.9%. At runtime, the reconstruction mean squared error of the detector is used with Inductive Conformal Prediction [27] and power martingale [28] to compute a martingale value. We compared the martingale value with a threshold of 20, which was empirically selected for OOD detection.

4) Risk Score: To compute S_{Risk}, we first compute the ReSonAte score using the BTD with the hazard of "roadway obstruction" shown in Fig. 2. To compute the collision rate λ, we analyze the conditional relationships of BTD barriers and threats using the calibration set. The probability function of barriers $B1$ and $B2$ are dependent on the continuous-valued output of the OOD detector and the binary state of the anomaly detectors, which we capture using the following equation.

$$f_b(x = (B_1, B_2), s) = (1 - P(x|s.m.LEC)) \cdot \prod_{d \epsilon S^{B2}} \frac{P(x|d)}{0.4} \cdot f_s$$
$$P(x|s.m.LEC) = (1 + e^{-0.049 \cdot (t.m.LEC - 5.754)})^{-1} \quad (2)$$

Where, $P(x|s.m.LEC)$ is a sigmoid function used to capture the continuous values of the OOD detector, the discrete-valued probability of the anomaly detectors that constitutes the state-variables in S^{B2}. In our example, $B1$ and $B2$ were affected by the image quality, which we detect using the blur and occlusion detectors whose probabilities are reported in Fig. 2. Also, f_s is the failure rate of the sensors, which is assumed to be constant for both the camera and the radar sensors on which the barriers are dependent. Further, the probability function of $B3$ is conditionally dependent on the precipitation levels and the operation of the radar as shown in Fig. 2. These conditional relationships were inferred by clustering the calibration scenes based on the precipitation levels. In a similar approach, we also estimated the frequency of threat occurrence. In this example, the frequency of threats $T1$ and $T2$ depended on the road segments. An intersection had a higher frequency of threats than a side road. Finally, we use these probabilities to compute the dynamic collision rates and the ReSonAte score as discussed in Section IV-B.

Next, we compute the infraction score using $IS = 0.7 \cdot I_S + 0.8 \cdot I_R + I_{RD}$. Where I_S is a stop sign infraction, I_R is a red-light infraction, and I_{RD} is the deviation in the route taken by the AV. This score is added to the ReSonAte score to compute the S_{Risk}. Finally, we compute the risk threshold δ to be 0.65, as discussed in Section IV-B.

B. Baselines and Comparison Metrics

We compare our sampler to state-of-the-art baselines. (1) Random Search is a technique that samples the scene variables uniformly at random from their respective distributions. (2) Grid Search is a technique that exhaustively samples all the combinations of the scene variables in a given grid. (3) Halton Sequence Search [16] is a pseudo-random technique that samples the scene variables using co-prime as its bases. We compare these baselines using the following metrics, which align with the sampler objectives discussed in Section III.

1) Total Risk Scenes (TRS): The proportion of the high-risk scenes sampled to the total scenes sampled (N).

$$TRS(\%) = \frac{1}{N} \sum_{i=1}^{N} s_i \quad \text{if} \quad S^i_{Risk} > \delta \quad (3)$$

Where S^i_{Risk} is the risk across the i^{th} sampled scene, and δ is the risk threshold computed across the calibration set.

2) Diversity (D): We measure the coverage diversity using two cluster-based quantities. (1) the number of optimal clusters that can be generated from the N sampled scenes, as measured using silhouette score [21]. (2) the risk variance across each scene across all the clusters. These quantities are combined into computing the diversity score as follows.

$$Diversity = \sigma^2\{\frac{1}{len(\mathcal{C})}(\sum_{i=1}^{len(\mathcal{C})} S^i_{Risk}) | \forall \mathcal{C} \in N\} \quad (4)$$

Where $\mathcal{C}$ is a cluster of scenes generated on the scene variables. S^i_{Risk} is the risk of the i^{th} scene in a cluster. To select the optimal number of clusters, we perform k-means clustering [29] using silhouette score as the selection metric.

3) Search Time: The overall time taken by the sampler to sample N scenes and execute them in the simulator.

C. Results

To compare our samplers against the baselines, we executed each sampler for 250 iterations. We started the samplers at the same initial condition with time-of-day = 45° (noon) and a value of zero for the other variables. Also, instead of starting the GBO sampler from scratch, we provide it a "warm start" using the results from the random sampler. Further, we empirically selected the following parameters for controlling the exploration vs. exploitation tradeoff. (1) Number of neighbors K = 6, and threshold τ = 10 for the RNS sampler, and (2) β = 30 for the UCB function of the GBO sampler. The simulation operates at a fixed rate of 20 Frame Per Seconds in these experiments. Fig. 6 shows the qualitative comparison of the scenes sampled by the samplers. The random and Halton samplers explore the search space, but they cannot exploit it because of their passive search strategy. On the other hand, the grid sampler orderly exploits every point in the grid, limiting its exploration. In comparison, the

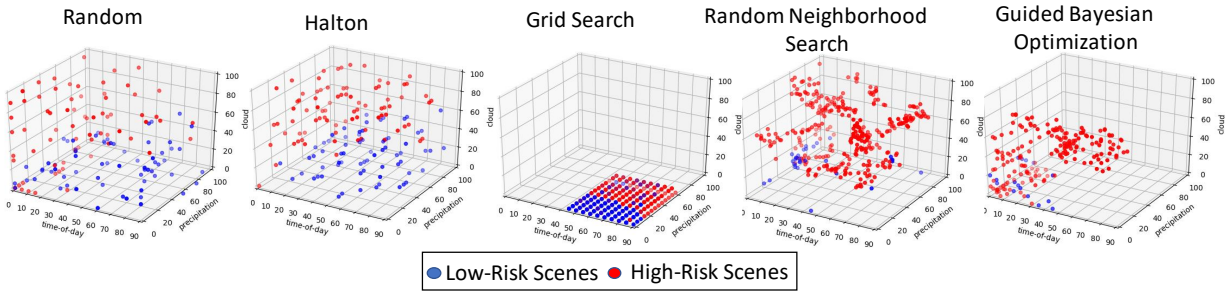

Fig. 6: Comparison of the 250 scenes sampled by the different samplers. While random and Halton samplers only explore the space, grid sampler only exploits. Our samplers balance the exploration vs. exploitation, which is evident from the dense scene clusters in different regions of the search space.

Sampler	Total Risk Scenes (%)	Diversity			Search Time (mins)
		Cluster Selection		Diversity Score	
		# of Clusters	Silhouette score		
Random	66	3	0.34	0.02	323
Halton	71	2	0.27	0.07	315
Grid	56	2	0.71	0.135	309
RNS	83	6	0.56	0.193	332
GBO	92	4	0.62	0.632	897

TABLE II: Comparing the sampling approaches using the comparison metrics.

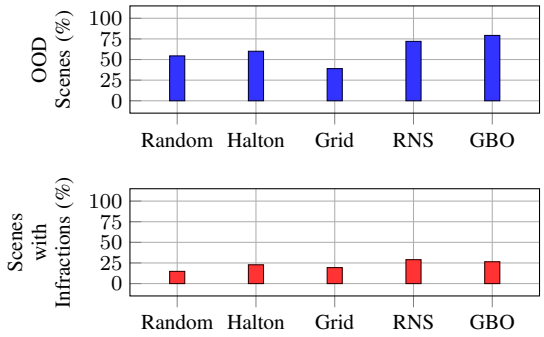

Fig. 7: (Top) Total OOD scenes and (Bottom) Total scenes with infractions sampled among the total 250 scenes. Our samplers sampled a higher percent of OOD scenes and scenes with infractions.

RNS and GBO samplers performs balanced exploration and exploitation. This is evident from the dense scene clusters at different regions of the search space. Further, we compared the samplers using the comparison metrics.

Total Risk Scenes: Table II lists the total risk scenes generated by different samplers. As seen, the random, Halton, and grid samplers sampled 66%, 71%, and 56% of high-risk scenes, respectively. In comparison, the RNS sampler sampled 83% and GBO sampled 92% high-risk scenes among the 250 scenes that were sampled. Also, Fig. 7 illustrates the number of OOD scenes and scenes with infractions that were generated by the samplers in 250 iterations. As seen, the GBO and the RNS samplers generated higher OOD scenes compared to the baselines. Also, the RNS sampler sampled the highest scenes with infractions followed by GBO and Halton samplers. The actual infractions are low even when the risk is high for the following reasons. (1) the LEC controller was sufficiently trained to avoid AV infractions, (2) the AEBS worked sufficiently well to stop the AV.

We also started the simulator at 5 different initial conditions.

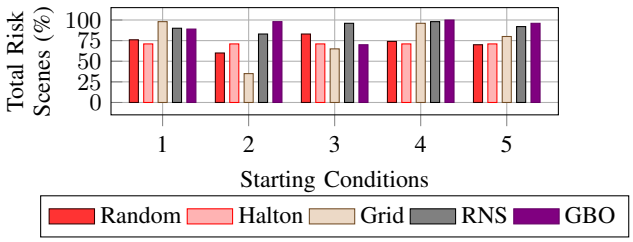

Fig. 8: Total high risk scenes sampled by samplers starting from different initial conditions. Generally, our samplers sampled higher percent of risky scenes across all the starting conditions.

(1) T = 0° (dusk), (2) T = 45° (noon), (3) T = 90° (noon), (4) T = 45° (noon), P = 50%, C = 50%, and (5) T = 45°, P = 0%, C = 50%, and the other variables were set to zero. Each sampler was executed for 50 iterations from these initial conditions. Fig. 8 shows the high-risk scenes generated by the different samplers. The RNS and GBO samplers generated high-risk scenes across the different starts as compared to the baselines. Among the baselines, the grid search generated higher high-risk scenes than the other two samplers. To note, different stating conditions did not affect the Halton sampler.

Diversity: To compute diversity, we classified the sampled scenes into clusters by applying k-means clustering analysis along with silhouette score. As seen in Table II, the scenes generated by the RNS and GBO samplers could be classified into a higher number of clusters (6 and 4) as compared to the Halton, grid, and random samplers that could only be classified into 2, 2, and 3 clusters, respectively. We used these clusters to compute the diversity score in Eq. (4). As seen, the GBO and RNS samplers had a higher diversity score of 0.63 and 0.19 as compared to the Halton, random, and grid samplers, which had scores of 0.07, 0.02, and 0.13, respectively. The GBO and RNS samplers had higher coverage diversity because they sampled a higher number of scenes around a scene that was previously found to be of high risk to the AV.

Search Times: We report the search times of the samplers in Table II. The random, Halton, grid, and RNS samplers have comparable search times of 323, 315, 309, and 332 minutes, respectively. However, the GBO sampler required the highest search time of 897 minutes because of its scalability issue discussed in Section IV-C2. Further, for a direct comparison, we counted the number of scenes sampled by each sampler within a fixed budget of 60 minutes. The random, Halton, grid

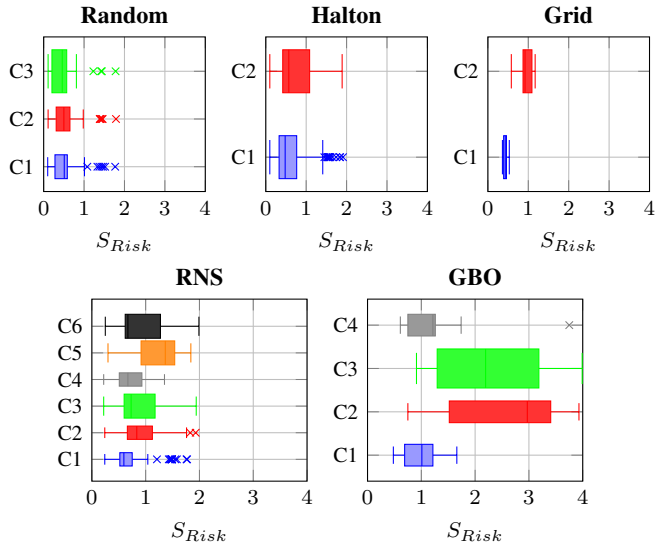

Fig. 9: The variance of S_{Risk} across scene clusters generated by different samplers. Our samplers generate more clusters with a high variance of S_{Risk}. Y-axis shows the number of clusters, and the Y-axis shows the S_{Risk} score.

and RNS samplers sampled 47, 51, 49, and 46 scenes in 60 minutes. But, the GBO sampler only sampled 19 scenes.

D. Discussion

The key takeaways from the experiments are. First, we observe that the RNS and GBO samplers outperform the baselines by generating a higher percentage of high-risk scenes (first objective). Second, we observe that the scene generated by the RNS and GBO samplers have a higher diversity score as compared to the baselines (second objective). Third, the GBO sampler is the most effective in generating high-risk scenes, and its informed exploration results in diverse search. However, the use of GP model increases the search time, which we plan to address in the future using a scalable GP model. In comparison, the RNS sampler is efficient and requires lower search times to generate high-risk scenes.

Lastly, the high-risk scenes identified by our samplers can be used to improve the system's LECs. To illustrate, the AV system in our case study was susceptible to scenes from dusk with high precipitation ($T < 45$ and $P > 50$) as identified by our samplers. So, we trained a new LEC controller model $M_{retrain}$ on a dataset that contained both the original train scenes and 50 high-risk scenes sampled by the RNS sampler. We trained the model for 50 epochs and saved several model weights, and utilized the weights that optimized the AV's driving proficiency. We then compared the performance of the original model M_{orig} (model trained on original training scenes) and the retrained model $M_{retrain}$ in terms of the actual system infractions, which included collision, running a red light or stop sign, and route deviation. The comparison is shown in Fig. 10, and for this, we curated 80 operational scenes that included random scenes and scenes where the system previously failed. We observed that $M_{retrain}$ had only 6 scenes with infractions as compared to 11 scenes of M_{orig}. Retraining the model on the high-risk scenes reduced

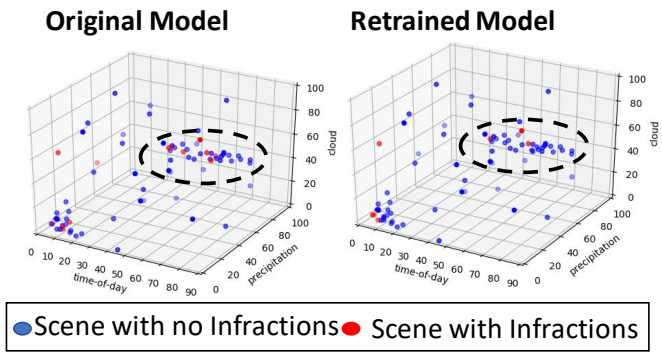

Fig. 10: Performance of $M_{retrain}$ and M_{orig} models. $M_{retrain}$ model had fewer infractions, especially in the scenes from the region highlighted.

the system infractions, especially those from the highlighted region where it previously had higher infractions. Additionally, we observed that $M_{retrain}$ had improved driving proficiency with fewer route deviations for the high-risk scenes. However, both models performed several red-light infractions in general, and retraining did not address the problem.

VI. Related Work

Tools like Dakota [9] with sampling approaches like incremental sampling, importance sampling, and adaptive sampling have been long available for uncertainty quantification in engineering design. Such samplers have recently gained interest for sampling simulation-based scenes in the AV domain. Several domain-specific SDLs like Scenic [8] and MSDL [13] have been designed for specifying scenarios of complex traffic conditions and operating conditions to identify counterexamples that affect the system's safety. These languages use probabilistic samplers that sample scenes across the entire search space formed by the combination of scene variables. For example, Grid search is a popular passive sampler, which takes a single risky scene as the starting condition and generates similar risky scenes around it. However, manual tuning makes the grid sampler labor-intensive and time-consuming. For accelerating the search process, an alternate approach [10] uses importance sampling to identify important scene variables and sample scenes using them. Worst-Case Scenario Evaluation [5] is another approach that uses model-based optimization to identify the weakness of the system and uses this information to generate worst-case disturbances. VERIFAI [11] software toolkit has several passive samplers like random search, Halton search [16], and active samplers like cross-entropy optimization and Bayesian Optimization. However, these samplers do not effectively balance the exploration vs. exploitation trade-off. Viswanadha, Kesav, *et al.* [12] recently proposed a multi-armed bandit sampler that balances the two strategies.

Also, recently, there has been growing interest in using generative models for generating risky scenes [6], [30]–[32]. In this approach, scenes are randomly sampled from a distribution learned by a generative model instead of sampling from the entire search space (as performed by the samplers discussed above). For example, Ding, Wenhao, *et al.* [6] use an autore-

gressive model to factorize the scene variables into conditional probability distributions, which are then sampled to generate risky traffic scenes. In a prior work [30], they have also used a Variational Autoencoder to project high-dimensional traffic information into a lower-dimensional latent space, which is sampled to generate critical traffic scenes. Vardhan *et al.* [31] uses a Gaussian Mixture Model to find the corner case scenes in the training set that is not well learned by the system. Another approach [32] uses generative adversarial imitation learning to perform adaptive importance-sampling to learn rare events from an underlying data distribution.

VII. Conclusion

We presented a scene generation approach that integrates a Scenario Description Language with two samplers and a risk-based scoring function for generating high-risk scenes. Random Neighborhood Search and Guided Bayesian Optimization are the proposed active samplers that perform constraint-based sampling and balance the exploration vs. exploitation to guide them towards sampling clusters of high-risk scenes. We applied these samplers to an AV case study. Our evaluations show that the proposed samplers could sample a higher number of high-risk scenes that could be clustered into a higher number of clusters than the conventional baselines.

Acknowledgement: This work was supported by the DARPA Assured Autonomy project and Air Force Research Laboratory. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA or AFRL.

References

[1] P. Bishop and R. Bloomfield, "A methodology for safety case development," in *Safety and Reliability*. Taylor & Francis, 2000.

[2] E. Denney, G. Pai, and I. Habli, "Dynamic safety cases for through-life safety assurance," in *IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 2. IEEE, 2015, pp. 587–590.

[3] V. K. Sundar, S. Ramakrishna, Z. Rahiminasab, A. Easwaran, and A. Dubey, "Out-of-distribution detection in multi-label datasets using latent space of β-vae," *arXiv preprint arXiv:2003.08740*, 2020.

[4] C. Hartsell, S. Ramakrishna, A. Dubey, D. Stojcsics, N. Mahadevan, and G. Karsai, "Resonate: A runtime risk assessment framework for autonomous systems," in *International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, may 2021.

[5] W.-H. Ma and H. Peng, "A worst-case evaluation method for dynamic systems," 1999.

[6] W. Ding, B. Chen, M. Xu, and D. Zhao, "Learning to collide: An adaptive safety-critical scenarios generating method," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 2243–2250.

[7] S. Khan, M. Hayat, S. W. Zamir, J. Shen, and L. Shao, "Striking the right balance with uncertainty," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 103–112.

[8] D. J. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, "Scenic: A language for scenario specification and scene generation," in *Proceedings of the 40th annual ACM SIGPLAN conference on Programming Language Design and Implementation (PLDI)*, June 2019.

[9] K. Dalbey, M. Eldred, G. Geraci, J. Jakeman, K. Maupin, J. A. Monschke, D. Seidl, A. Tran, F. Menhorn, and X. Zeng, "Dakota a multilevel parallel object-oriented framework for design optimization parameter estimation uncertainty quantification and sensitivity analysis: Version 6.14 theory manual." Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), Tech. Rep., 2021.

[10] D. Zhao, X. Huang, H. Peng, H. Lam, and D. J. LeBlanc, "Accelerated evaluation of automated vehicles in car-following maneuvers," *IEEE Transactions on Intelligent Transportation Systems*, 2017.

[11] T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, H. Ravanbakhsh, M. Vazquez-Chanlatte, and S. A. Seshia, "Verifai: A toolkit for the formal design and analysis of artificial intelligence-based systems," in *International Conference on Computer Aided Verification*. Springer, 2019, pp. 432–442.

[12] K. Viswanadha, E. Kim, F. Indaheng, D. J. Fremont, and S. A. Seshia, "Parallel and multi-objective falsification with scenic and verifai," in *International Conference on Runtime Verification*. Springer, 2021.

[13] O. foretellix, "Open m-sdl." [Online]. Available: https://www.foretellix.com/open-language/

[14] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," *arXiv:1711.03938*, 2017.

[15] J. Jerebic, M. Mernik, S.-H. Liu, M. Ravber, M. Baketarić, L. Mernik, and M. Črepinšek, "A novel direct measure of exploration and exploitation based on attraction basins," *Expert Systems with Applications*, vol. 167, p. 114353, 2021.

[16] J. H. Halton, "On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals," *Numerische Mathematik*, vol. 2, no. 1, pp. 84–90, 1960.

[17] O. CARLA, "Carla leaderboard challenge," 2020. [Online]. Available: https://leaderboard.carla.org/

[18] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," in *Conference on Robot Learning*. PMLR, 2020, pp. 66–75.

[19] F. Cai and X. Koutsoukos, "Real-time out-of-distribution detection in learning-enabled cyber-physical systems," in *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*. Los Alamitos, CA, USA: IEEE Computer Society, 2020. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICCPS48487.2020.00024

[20] R. Ferdous, F. Khan, R. Sadiq, P. Amyotte, and B. Veitch, "Analyzing system safety and risks under uncertainty using a bow-tie diagram: An innovative approach," *Process Safety and Environmental Protection*, vol. 91, no. 1-2, pp. 1–18, 2013.

[21] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[22] I. Dejanović, R. Vaderna, G. Milosavljević, and Ž. Vuković, "Textx: a python tool for domain-specific languages implementation," *Knowledge-Based Systems*, vol. 115, pp. 1–4, 2017.

[23] J. H. Friedman, J. L. Bentley, and R. A. Finkel, *An algorithm for finding best matches in logarithmic time*. Department of Computer Science, Stanford University, 1975.

[24] P. Auer, "Using confidence bounds for exploitation-exploration tradeoffs," *Journal of Machine Learning Research*, vol. 3, no. Nov, 2002.

[25] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, "When gaussian process meets big data: A review of scalable gps," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 11, pp. 4405–4423, 2020.

[26] J. L. Pech-Pacheco, G. Cristóbal, J. Chamorro-Martinez, and J. Fernández-Valdivia, "Diatom autofocusing in brightfield microscopy: a comparative study," in *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, vol. 3. IEEE, 2000, pp. 314–317.

[27] G. Shafer and V. Vovk, "A tutorial on conformal prediction," *Journal of Machine Learning Research*, vol. 9, no. Mar, pp. 371–421, 2008.

[28] V. Fedorova, A. Gammerman, I. Nouretdinov, and V. Vovk, "Plug-in martingales for testing exchangeability on-line," *arXiv preprint arXiv:1204.3251*, 2012.

[29] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl *et al.*, "Constrained k-means clustering with background knowledge," in *Icml*, vol. 1, 2001.

[30] W. Ding, W. Wang, and D. Zhao, "A new multi-vehicle trajectory generator to simulate vehicle-to-vehicle encounters," *arXiv preprint arXiv:1809.05680*, 2018.

[31] H. Vardhan and J. Sztipanovits, "Rare event failure test case generation in learning-enabled-controllers," in *2021 6th International Conference on Machine Learning Technologies*, 2021, pp. 34–40.

[32] M. O'Kelly, A. Sinha, H. Namkoong, J. Duchi, and R. Tedrake, "Scalable end-to-end autonomous vehicle testing via rare-event simulation," *arXiv preprint arXiv:1811.00145*, 2018.