# From Particles to Perils: SVGD-Based Hazardous Scenario Generation for Autonomous Driving Systems Testing

LINFENG LIANG, Macquarie University, Australia
XIAO CHENG, Macquarie University, Australia
TSONG YUEH CHEN, Swinburne University of Technology, Australia
XI ZHENG*, Macquarie University, Australia

Simulation-based testing of autonomous driving systems (ADS) must uncover realistic and diverse failures in dense, heterogeneous traffic. However, existing search-based seeding methods (e.g., genetic algorithms) struggle in high-dimensional spaces, often collapsing to limited modes and missing many failure scenarios. We present PtoP, a framework that combines adaptive random seed generation with Stein Variational Gradient Descent (SVGD) to produce diverse, failure-inducing initial conditions. SVGD balances attraction toward high-risk regions and repulsion among particles, yielding risk-seeking yet well-distributed seeds across multiple failure modes. PtoP is plug-and-play and enhances existing online testing methods (e.g., reinforcement learning–based testers) by providing principled seeds. Evaluation in CARLA on two industry-grade ADS (Apollo, Autoware) and a native end-to-end system shows that PtoP improves safety violation rate (up to 27.68%), scenario diversity (9.6%), and map coverage (16.78%) over baselines.

CCS Concepts: • **Software and its engineering → Search-based software engineering**.

Additional Key Words and Phrases: Autonomous driving systems, Software testing

**ACM Reference Format:**
Linfeng Liang, Xiao Cheng, Tsong Yueh Chen, and Xi Zheng. 2026. From Particles to Perils: SVGD-Based Hazardous Scenario Generation for Autonomous Driving Systems Testing. *Proc. ACM Softw. Eng.* 3, FSE, Article FSE146 (July 2026), 22 pages. https://doi.org/10.1145/3808153

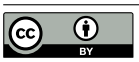
## 1 Introduction

Autonomous driving systems (ADS) [1, 10, 45] are safety-critical cyber-physical systems that require rigorous testing to ensure reliability prior to deployment [6, 8]. While real-world testing provides valuable insights, it is costly and time-consuming [29], making simulation-based testing [12, 13, 17, 35] a practical alternative for evaluating ADS across diverse scenarios. To be effective, however, simulation must bridge the gap to real-world conditions by moving beyond simplified assumptions and uncovering safety violations in realistic settings with *dense traffic and complex interactions among heterogeneous dynamic objects* (e.g., vehicles, cyclists, and pedestrians) [4, 5].

Instead of generating pre-defined trajectories for dynamic objects—which is often unrealistic and exacerbates the simulation-to-reality gap—some latest approaches [35] adopt a two-stage process (1) *Offline seeding*—generating initial states of dynamic objects (e.g., spawn positions, velocities, and orientations of vehicles, cyclists, and pedestrians) relative to the ego vehicle; and (2) *Online testing*—the dynamic update of object trajectories based on complex interactions with other participants in

*Corresponding author

Authors' Contact Information: Linfeng Liang, Macquarie University, , Australia, linfeng.liang@hdr.mq.edu.au; Xiao Cheng, Macquarie University, , Australia, xiao.cheng@mq.edu.au; Tsong Yueh Chen, Swinburne University of Technology, , Australia, tychen@swin.edu.au; Xi Zheng, Macquarie University, , Australia, james.zheng@mq.edu.au.




ACM 2994-970X/2026/7-ARTFSE146
https://doi.org/10.1145/3808153

the scenario, including the ego vehicle, governed by predefined policies or traffic models. These updates are commonly achieved through approaches such as reinforcement learning (RL)-based methods [35] or gradient-based methods [23]. Nevertheless, the quality of the offline seeding stage remains highly crucial: small perturbations to initial conditions can yield substantially different scenario evolutions and thus alter the likelihood of failures.

Seeding is commonly cast as a search problem and addressed with search-based techniques such as genetic algorithms (GAs) [13, 20, 32, 35, 46]. However, under dense heterogeneous traffic, such methods face two challenges: *effectiveness*—identifying rare, failure-inducing configurations; and *diversity*—covering a wide range of distinct failure modes. These stem from the *high-dimensional search space* induced by multiple object types, where failure cases cluster in disjoint regions. Classical GA operators often converge prematurely and struggle to preserve diversity, leading to violations concentrated in a few clusters. For example, in the TOWN04 map of CARLA [18], 372 spawn points with just five surrounding vehicles yield $2.545 \times 10^{15}$ possible initial configurations. Adding continuous parameters (e.g., headings, speeds) and other agents like pedestrians further expands the space, making exhaustive or unguided search infeasible.

To address the seeding challenges while remaining agnostic to online testing, we present PToP, a general-purpose ADS testing framework that generates diverse initial conditions while seamlessly integrating with existing online testing tools [23, 35] to enhance the overall performance. A seed pool is maintained by an adaptive random seed generator (ARSG), building on adaptive random testing [11], which samples candidates that maximize their minimum distance to executed seeds to preserve diversity. These candidates are then refined using Stein Variational Gradient Descent (SVGD) [36], which we adapt and re-implement within our framework to approximate a posterior over the initial states from ARSG: the prior encodes plausible initial conditions, the likelihood quantifies the propensity of a condition to induce a failure, and the posterior concentrates on high-risk regions while maintaining coverage. Each **particle** is a parameter vector that describes a dynamic object's initial state; SVGD applies gradient attraction toward high-risk regions and kernel repulsion to spread out (prevent collapse), uncovering multiple disjoint failure clusters in the high-dimensional search spaces and addressing GA's premature convergence. Refined seeds are fed back to the seed pool and simultaneously passed to the online tester to instantiate scenarios and trigger safety violations.

The online tester can be any existing online testing method, such as RL-based methods [35] and gradient-based methods [23]. This is because PToP is designed as a **plug-and-play** ADS testing framework that treats existing online testing methods as plug-ins. In PToP, the *offline seeder* is responsible for generating high-quality and diverse initial seeds that define the starting states of dynamic objects. The *online tester* then utilizes these seeds to produce concrete trajectories through simulation, exposing ADS safety issues under various and realistic conditions. They interface via gradient signals supplied by a learning-based hazard model that is trained using the online interaction data collected from the online tester to estimate failure likelihood from initial conditions. Importantly, this process is iterative: feedback from the online stage guides subsequent seeding, progressively improving both the coverage of failure scenarios and the fidelity of the testing pipeline.

In summary, our main contributions are as follows:

- We introduce PToP, a plug-and-play ADS testing framework for dense, heterogeneous traffic that augments existing online testers to improve failure-finding effectiveness and diversity.
- We propose an SVGD-based seeding method that treats seeding as posterior inference—using a prior over plausible states, a learned hazard likelihood trained with online testing, and SVGD updates (gradient attraction + kernel repulsion) to target multiple high-risk regions while preserving diversity.

- We empirically evaluate PtoP in CARLA across multiple ADS and testers, showing consistent gains in failure-finding effectiveness and coverage: it uncovers substantially more safety violations and a wider variety of failure modes, improving baseline testers rather than replacing them.

## 2 Related Work

### 2.1 ADS Testing via Offline Seed Generation

Substantial effort has been devoted to testing ADS using offline seed generation followed by online scenario execution in simulation. In this setting, initial test configurations are generated offline, and the scenarios are executed in a simulator where dynamic objects evolve during runtime, but their trajectories remain pre-defined, limiting realism and adaptability.

*Data-driven* approaches [14, 22, 24, 43, 44] leverage deep learning pipelines to synthesize realistic initial scenes and agent behaviors from real-world datasets. These methods excel at producing diverse, plausible simulation environments for training and evaluation; however, they largely emphasize open-ended scene generation and do not explicitly target the systematic triggering of safety violations in deployed ADSs.

*Combinatorial* methods [9, 27, 33] systematically enumerate combinations of input parameters or features to ensure broad coverage, but rely exclusively on offline techniques—such as ontology-based modeling and pairwise/interaction-based sampling.

*Search-based* approaches [2, 7, 12, 13, 19, 25, 26, 26, 32, 35, 39, 41] use evolutionary algorithms (e.g., GA) to randomly generate chromosome representations, execute them in a simulator, and employ multi-objective fitness functions to select high-fitness parent scenarios; crossover and mutation then yield offspring with potentially higher risk. While effective at identifying safety violations, these methods often exhibit limited diversity in the violations they uncover due to the inherent convergence behavior of evolutionary search.

PtoP is a search-based testing framework that combines offline seed generation with online scenario exploration. Unlike prior approaches that rely solely on pre-defined or GA-based seeding, PtoP integrates adaptive random seed generation with an SVGD-guided refinement, enabling both diversity and targeted exploration of high-risk regions. This design allows PtoP to expose a broader range of safety violations while ensuring coverage across heterogeneous traffic scenarios.

### 2.2 ADS Online Testing

We use the term *online testing* to refer to methods where dynamic objects' trajectories are generated and adapted on the fly during simulation through interactions with the ego vehicle and environment. This differs from testing using only offline seed generation, where initial conditions are fixed in advance and trajectories remain pre-defined. Online testing approaches [20, 23, 28, 35, 37, 38] employ reinforcement learning or gradient-based optimization to manipulate dynamic objects or environmental factors (e.g., weather) in real time within an episode.

Online testing methods are effective, but without failure-inducing initial scenarios the search space remains enormous. PtoP addresses this challenge by combining adaptive random testing (ART) for diverse seed generation with an SVGD-guided refinement that concentrates seeds in high-risk regions. The resulting scenarios are then passed to the online testing component for interaction-based exploration, enabling the discovery of more numerous and diverse safety violations in ADS.

## 3 PtoP Approach

This section presents the design of PtoP. We first provide an overview of the framework, followed by detailed descriptions of its offline seeding and online testing.

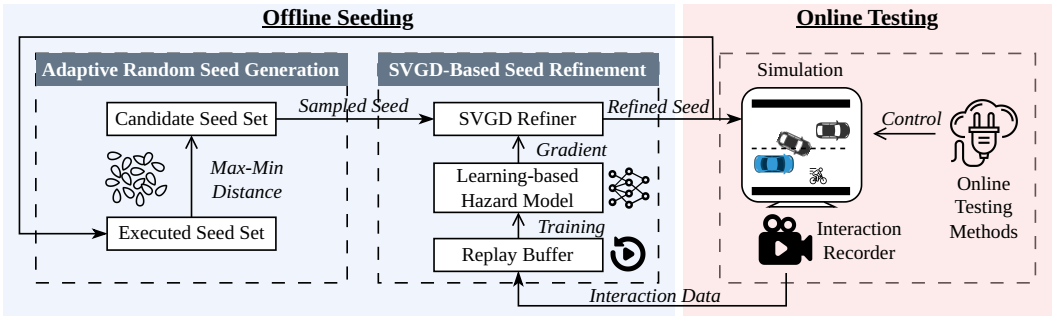



Fig. 1. Framework overview of PTOP.

### 3.1 Overview

Figure 1 presents the overall framework of PTOP, which comprises an offline seeder and an online tester that operate iteratively to improve one another. The offline seeder consists of two stages, Adaptive Random Seed Generation (ARSG, §3.2) and SVGD-based Seed Refinement (§3.3), that jointly produce diverse, failure-inducing test seeds. The online tester (§3.4) executes these seeds in a simulator using established online testing algorithms (e.g., reinforcement learning and gradient-based search) and returns hazardous-scenario feedback to the offline seeder.

The process begins with ARSG, which selects a seed from a candidate pool to maximize diversity relative to previously executed seeds. The selected seed is then refined via SVGD, which updates the associated particle ensemble to generate initial conditions for dynamic objects that are both more diverse and more likely to induce failures. Next, the refined seed is passed to the online tester, which manipulates the dynamic objects to elicit safety violations; simultaneously, the seed is also added to the set of executed seeds maintained by ARSG. During each episode, an interaction recorder logs interactions (e.g., speed, positions and headings) between each dynamic object and the ego vehicle and appends them to a replay buffer. Upon completion of the episode, the hazard model is updated using the replay buffer and its gradients are supplied to the SVGD refiner to guide the refinement of the next seed.

### 3.2 Adaptive Random Seed Generation

*3.2.1 Seed Modeling.* To systematically encode the initial conditions of a test scenario, we adopt a chromosome-based representation. This representation comprises two primary components: the ego vehicle's route gene and the dynamics gene. The ego vehicle's route gene encodes the ego vehicle's initial orientation and position, as well as its destination, which is defined as the farthest valid waypoint within 200 meters along the vehicle's initial heading. This approach ensures consistency in test case length and comparability across scenarios. The dynamics gene specifies the type of each dynamic object present in the scenario, together with their respective initial positions and orientations. This structured encoding facilitates systematic scenario generation and supports comprehensive exploration of the scenario space.

*3.2.2 Adaptive Random Seed Generator.* To facilitate comprehensive exploration of the scenario space, we propose the Adaptive Random Seed Generator (ARSG), which is designed to generate highly diverse test seeds. The ARSG adopts the Adaptive Random Testing (ART) strategy [11], which systematically enhances diversity among generated seeds. Specifically, ARSG maintains a set of candidate seeds and, at each iteration, selects the candidate that **maximizes the minimum distance to all previously executed seeds**. Once a seed is selected and executed in the simulator, it is added to the set of executed seeds.

Formally, let the set of executed seeds be

$$T = \{t_1, t_2, \ldots, t_n\}$$

and the set of candidate seeds be

$$C = \{c_1, c_2, \ldots, c_k\}.$$

The ARSG selects the next seed $c^*$ according to

$$c^* = \arg\max_{c_j \in C} \left( \min_{t_i \in T} \text{dist}(c_j, t_i) \right),$$

where $\text{dist}(\cdot, \cdot)$ denotes a distance metric, typically Euclidean distance. The candidate seeds in $C$ are generated randomly.

The motivation for this approach stems from the observation that purely random seed generation disregards the spatial distribution of seeds within the input domain, which can result in suboptimal failure detection—especially for failures that manifest in non-point patterns [11]. By contrast, the ARSG explicitly promotes uniform spatial coverage, thereby increasing the probability of intersecting failure-prone regions that may be clustered or structured (e.g., as strips or blocks) within the input space. This uniformity is particularly advantageous for the subsequent seed refinement stage, as it provides a diverse and well-dispersed set of initializations, helping to avoid premature convergence to local optima and enabling more effective exploration of high-risk regions.

### 3.3 SVGD-Based Seed Refinement

To effectively generate failure-inducing scenarios while maintaining diversity, we introduce a seed refinement approach based on Stein Variational Gradient Descent (SVGD). This approach integrates an interaction recorder and a replay buffer (§3.3.1) to systematically capture and store online interaction data from the online tester (§3.4). Leveraging this data, we train a learning-based hazard model (§3.3.2) that estimates the risk associated with different initial scenario configurations. The gradients provided by the trained hazard model are then utilized to guide the SVGD-based seed refiner (§3.3.3) toward regions of the scenario space with elevated risk, thereby enhancing the probability of uncovering hazardous cases. Simultaneously, the kernel-based repulsion mechanism inherent to SVGD fosters exploration across multiple modes and diverse regions of the scenario space, ensuring comprehensive and balanced coverage.

*3.3.1 Interaction Recorder and Replay buffer.* The interaction recorder and replay buffer are designed to capture and store online interaction data between the ego vehicle and surrounding dynamic objects during online testing. At each simulation frame $t$, the system records the speed ($v_t^{\text{ego}}$ for the ego vehicle and $v_t^{(i)}$ for each dynamic object $i$), positions ($p_t^{\text{ego}}$ and $p_t^{(i)}$), and headings ($\psi_t^{\text{ego}}$ and $\psi_t^{(i)}$) of all relevant agents. This set of state information is continuously appended to the replay buffer, thereby facilitating subsequent analysis and enabling the learning of hazard models from the recorded interactions.

*3.3.2 Learning-based Hazard Model.* To estimate the probability that a specific initial configuration of the ego vehicle and surrounding dynamic objects will result in a safety violation within a finite time horizon, we employ a *learning-based hazard model*. In the following, we first detail the construction of the feature vector that encodes the relative initial state between the ego vehicle and each dynamic object, and subsequently describe the derivation of learning targets from closed-loop simulation rollouts.

Formally, let $z \in \mathbb{R}^m$ denote a feature vector encoding the relative initial state between the ego vehicle and a dynamic object. For each dynamic object $i$ at the initial frame ($t$=0), we construct a

five-dimensional, ego-centric feature vector $z_i \in \mathbb{R}^5$ capturing relative geometry and lane context:

$$z_i = \Big[ \underbrace{\Delta s_i}_{\text{longitudinal}}, \underbrace{\Delta d_i}_{\text{lateral}}, \underbrace{\cos\Delta\psi_i}_{\text{heading 1}}, \underbrace{\sin\Delta\psi_i}_{\text{heading 2}}, \underbrace{\lambda_i}_{\text{lane overlap}} \Big].$$

Here, $(\Delta s_i, \Delta d_i)$ represent the longitudinal and lateral offsets of dynamic object $i$ in the ego frame at $t$=0, and $\Delta\psi_i = \psi^{(i)} - \psi^{\text{ego}}$ denotes the relative yaw. The variable $\lambda_i \in [0, 1]$ quantifies the degree to which dynamic object $i$ overlaps with the ego's current lane, where $\lambda_i = 1$ indicates perfect alignment within the same lane, and $\lambda_i = 0$ indicates complete absence from the ego's lane. All trigonometric features are expressed in radians; the use of both $\cos\Delta\psi_i$ and $\sin\Delta\psi_i$ ensures continuity and avoids angle wrap-around issues. Each normalized component is clipped to $[-1, 1]$ to enhance training stability. Notably, we omit kinematic features, as all dynamic objects are initially static.

*Learning targets from closed-loop rollouts.* Supervision based solely on collision events is excessively sparse for the driving domain: while most episodes do not result in collisions, many contain near-miss interactions that are highly informative for search. To address this, we design a dense, hand-crafted hazard target for each dynamic object, which serves as ground truth for online training of the lightweight hazard model. This hazard model provides smooth and informative gradients to guide SVGD. The design of the learning target adheres to three principles: (i) capturing both the relative position of the ego vehicle with respect to a dynamic object and the rate at which the ego is closing in; (ii) preserving a strict notion of failure for ego-responsible collisions; and (iii) aggregating risk over a short, critical time window to mitigate noise.

Given an episode of length $H$ with frames $t = 0, \ldots, H$, we extract the states of the ego and dynamic objects to construct an ego-centric time series. Let $d_t^{(i)} = \|p_t^{\text{ego}} - p_t^{(i)}\|_2$ be the Euclidean distance between the ego and dynamic object $i$. We define the *unit bearing* (line of sight direction) $\hat{r}_t^{(i)} = \frac{p_t^{(i)} - p_t^{\text{ego}}}{\|p_t^{(i)} - p_t^{\text{ego}}\|_2}$ and the instantaneous closing speed

$$v_t^{\text{close}}(i) = -\hat{r}_t^{(i)\top}\big(v_t^{(i)} - v_t^{\text{ego}}\big), \quad \text{so } v_t^{\text{close}}(i) \geq 0 \text{ indicates approach.}$$

Using the *unit* bearing isolates direction from range, making $v_t^{\text{close}}(i)$ dimensionally correct (m/s) and interpretable as the time derivative of distance along the line of sight.

We locate the most critical moment for dynamic object $i$ by $t_i^\star = \arg\min_t d_t^{(i)}$ and consider a short 5-frames window $\mathcal{W}_i = \{t_i^\star - w, \ldots, t_i^\star + w\}$. Within $\mathcal{W}_i$ we compute bounded, $[0, 1]$-normalized indicators:

$$s_{t,i}^{\text{dist}} = e^{-\bar{d}_i}, \qquad s_{t,i}^{\text{head}} = \frac{1-\cos\Delta\psi_t^{(i)}}{2}, \qquad s_{t,i}^{\text{close}} = \text{clip}\left(\frac{\bar{v}_i^{\text{close}}}{v_{\max}}, 0, 1\right).$$

Here, $\Delta\psi_t^{(i)}$ is the relative yaw between the ego and dynamic object $i$ at time $t$, $\bar{d}_i$ is the average distance over $\mathcal{W}_i$, $\bar{v}_i^{\text{close}}$ is the average closing speed over $\mathcal{W}_i$, and $v_{\max}$ is the maximum velocity observed in the scenario. Intuitively, $s^{\text{dist}}$ is the normalized distance between the ego vehicle and the dynamic object and increases as the pair draws closer; $s^{\text{head}}$ is the normalized relative heading between the ego vehicle and the dynamic object and increases when their headings are aligned; and $s^{\text{close}}$ is the clipped closing speed of the ego vehicle relative to the dynamic object and increases when the ego is moving toward the dynamic object.

Each indicator is treated as an independent "no-risk" factor, and their contributions are aggregated multiplicatively across the window. For numerical stability and compactness, we perform this

aggregation in log-space:

$$S_i = \exp\left( \sum_{t \in \mathcal{W}_i} \left[ \log\left(1 - \tilde{s}_{t,i}^{\text{dist}}\right) + \log\left(1 - \tilde{s}_{t,i}^{\text{head}}\right) + \log\left(1 - \tilde{s}_{t,i}^{\text{close}}\right) \right] \right),$$

where $\tilde{s}$ denotes the normalized cue values.

The resulting $S_i$ approximates the probability of *no hazard* for dynamic object $i$ over the window $\mathcal{W}_i$; consequently, $y_i = 1 - S_i$ serves as a smooth near-miss score for the episode. In the event of a collision, $y_i \approx 1$; for episodes with no significant interaction, $y_i \approx 0$. We retain all samples, including those with $y_i \approx 0$, to address class imbalance and improve model calibration. Each feature vector $z_i$ is paired with its corresponding label $y_i$ to form the training data for the hazard model. This protocol yields dense, differentiable supervision for near-misses, enabling the surrogate model to provide informative gradients for SVGD-based search at the onset of each episode.

We train a hazard model $h_\theta : \mathbb{R}^m \to [0, 1]$ that maps an ego-centric feature vector $z$ to a calibrated hazard score. The model is updated online after each episode using a binary cross-entropy loss between $h_\theta(z)$ and $y$.

The motivation for this design is that **SVGD updates particles via gradients**; thus, a differentiable hazard model is essential. A purely hand-crafted, non-differentiable proxy cannot provide gradients, and recomputing heuristic labels would necessitate re-running closed-loop simulations for each proposal, which is computationally prohibitive. The learned surrogate enables efficient, one-shot, differentiable hazard estimation at the beginning of each episode, thereby facilitating effective gradient-based search.

Classical threat metrics like TTC and THW work only in simple car-following settings and become undefined or unstable in common ADS-testing situations involving pedestrians, cyclists, lateral merges, or zero/negative closing speeds. They therefore cannot serve as a universal, differentiable training signal for our gradient-based pipeline. Our near-miss label captures the same underlying factors—distance, orientation, and closing speed—while remaining smooth, well-defined, and applicable across all interaction types, enabling stable gradient flow to the hazard model and SVGD. Thus, it is more suitable than traditional threat metrics for heterogeneous ADS testing.

*3.3.3 SVGD Refiner.* To generate diverse and failure-inducing offline seeds, we employ an SVGD refiner to refine the seeds produced by the Adaptive Random Seed Generator. SVGD is a deterministic, particle-based method that approximates a target distribution by iteratively transporting a finite set of particles. At each iteration, particles move along a learned velocity field derived from Stein's identity and a positive-definite kernel. This yields two complementary effects: an *attraction* that drives particles toward high-probability regions via the target's score, and a *repulsion* that spreads particles across multiple modes to preserve diversity.

In our setting, the failure-inducing initial *relative geometry* between the ego vehicle and dynamic objects forms a distribution. Given the learned hazard model $h_\theta$, our goal is to efficiently move a small set of candidate relative initial conditions toward failure-prone regions while maintaining diversity. Under our settings, a dynamic object is represented as a particle in SVGD, encoded by the mutable subset of the ego–dynamic-object relative features $x$:

$$x \;=\; (\Delta s, \Delta d, \Delta\psi) \in \Omega, \qquad \Omega \;=\; \{\, |\Delta s| \le D_s,\ |\Delta d| \le D_d,\ |\Delta\psi| \le \Psi_{\max} \,\},$$

where $\Delta s$ is the relative longitudinal distance between the ego vehicle and the dynamic object, $\Delta d$ is the relative lateral distance, and $\Delta\psi$ is the relative heading; $D_s$, $D_d$, and $\Psi_{\max}$ are map-specified constants, with $D_s$ denoting the longitudinal range of the map, $D_d$ the lateral range, and $\Psi_{\max}$ the maximum heading difference, which equals $\pi$. Following the SVGD view [36], we define a target density $\pi(x)$

$$\pi(x) \;\propto\; \exp\big(\tau\, h_\theta(z)\big)\, \mathbb{1}[x \in \Omega],$$

where $\tau > 0$ is a temperature to control the strength of the gradient. Intuitively, $\nabla_x \log \pi(x) = \tau, \nabla_x h_\theta(z)$ **pulls particles toward high-hazard modes (efficiency)**, whereas the SVGD kernel term **enforces spread (diversity)**.

Let $x_i{}_{i=1}^{N}$ be the particles (one per selected dynamic object). SVGD transports the empirical measure by the velocity field

$$\phi(x_i) \;=\; \frac{1}{N}\sum_{j=1}^{N}\Big[\, k(x_j, x_i)\, \underbrace{\nabla_x \log \pi(x_j)}_{\tau\, g_j} \;+\; \beta\, \nabla_{x_j} k(x_j, x_i)\Big],$$

where $g_j = \nabla_x h_\theta(z(x_j))$ and $\beta \in (0, 1]$ weights repulsion. We use an anisotropic RBF kernel [36]

$$k(x, x') \;=\; \exp\Big(-\tfrac{1}{h}\,\|x - x'\|_\Lambda^2\Big),$$

The bandwidth $h$ is set by the median heuristic on pairwise $\|x_i - x_j\|_\Lambda^2$. Each iteration performs

$$x_i \;\leftarrow\; \Pi_\Omega\Big(x_i \;+\; \varepsilon\, \phi(x_i)\Big),$$

with stepsize $\varepsilon$ and projection $\Pi_\Omega$ clipping to the box $\Omega$. Together, these effects attract particles to high-hazard regions while maintaining diversity.

At the start of each episode we score all dynamics using $h_\theta$ and select the top-$K$ as particles. After each SVGD step, we apply a *hard minimum-separation* guard in $(\Delta s, \Delta d)$: if any pair $(i, j)$ violates $\sqrt{(\Delta s_i - \Delta s_j)^2 + (\Delta d_i - \Delta d_j)^2} \geq r_{\min}$, we push them apart along their local line of centers and re-clip to $\Omega$, where $r_{\min}$ equal to the lane width. This complements the kernel repulsion with a physically motivated constraint, preventing unrealistically clustered initializations.

*Advantages of Our Offline Seeding Approach.* In contrast to single-trajectory gradient ascent, SVGD simultaneously evolves a *set* of candidate seeds, leveraging a principled repulsion mechanism that effectively balances exploitation—by directing particles toward high-hazard regions as indicated by $h_\theta$—and exploration—by maintaining coverage across multiple distinct modes. The integration of SVGD with ARSG further amplifies these benefits: while ARSG broadens the search to encompass a wider range of failure modes, SVGD enables more thorough exploration within each identified mode. This synergy is particularly valuable under stringent testing budgets, as it accelerates convergence toward diverse, failure-inducing initializations compared to random search or heuristic-based methods. Moreover, the incorporation of guard and box constraints ensures that all generated seeds correspond to physically plausible scenario configurations.

### 3.4 Online Testing

In our framework, *online testing* denotes the process of dynamically modifying the trajectories of dynamic objects *during simulation*, on a step-by-step basis, rather than relying on static, predefined trajectories. This approach enables the testing process to adapt in real time to the behavior of the system under test, as inputs are generated interactively within the simulation loop. To facilitate this, ProP offers a plug-in API that supports seamless integration with a variety of existing online testing methodologies.

As a representative instantiation, we implement a gradient-based online testing technique inspired by KING [23]. KING leverages a differentiable bicycle-kinematics model to compute gradients that guide the optimization of dynamic objects' trajectories, where even minor adjustments to steering or acceleration can significantly influence the probability of a collision. In our approach,

the hazard model serves as an effective source of gradient information for online trajectory optimization: by performing gradient ascent, control updates are iteratively applied to each dynamic object's position and orientation to maximize the predicted hazard value.

We formally define the control space as follows. For vehicles and bicycles, the control input is a two-dimensional tuple (throttle, steering), representing the incremental changes in throttle and steering angle, respectively. For pedestrians, the control is a two-dimensional tuple $(x, y)$, corresponding to the longitudinal and lateral displacements. Our implementation generally follows KING's design for online control; further details regarding the mapping from hazard values to final control actions are provided in the **Section 3 of the supplementary material**.

## 4 Experimental Evaluation

In this section, we aim to evaluate the effectiveness of PtoP in efficiently generating diverse safety violation scenarios under dense and heterogeneous traffic conditions. Specifically, we benchmark PtoP against three state-of-the-art ADS testing methods across multiple map configurations and two advanced autonomous driving systems. To further elucidate the contributions of individual components within PtoP, we conduct comprehensive ablation studies. Additionally, we perform a user study to systematically assess the realism of the safety violations identified by our approach.

### 4.1 Research Questions

Our evaluation aims to answer the following research questions:

- RQ1 **What is the generalizability of PtoP?** We examine whether PtoP can interoperate with various online testing methodologies and assess its ability to generalize across diverse scenarios and autonomous driving systems.
- RQ2 **How effective is PtoP in identifying ADS safety violations compared to state-of-the-art techniques?** We evaluate whether PtoP can discover a greater number and a wider variety of safety violations than existing approaches, while also considering the efficiency in terms of time budget.
- RQ3 **What is the impact of individual components on the overall performance of PtoP?** We analyze the contributions of the Adaptive Random Seed Generation and Stein Variational Gradient Descent components to the effectiveness of PtoP.
- RQ4 **How realistic are the safety violations identified by PtoP?** We assess the plausibility and realism of the safety violations detected by PtoP through a user study involving domain experts.

### 4.2 Experimental Setup

We conduct our experiments using the high-fidelity CARLA simulator [18], which provides comprehensive support for a range of autonomous driving systems (ADSs) and enables the simulation of both urban and rural traffic environments [18, 47]. While our approach does not impose a theoretical upper bound on the number of dynamic objects, practical constraints such as computational resources and map dimensions lead us to configure each scenario with a fixed set of 20 dynamic objects, consisting of vehicles, bicycles, and pedestrians. Each experimental configuration is executed for 400 runs, with the entire process repeated four times to ensure statistical robustness; we report the averaged results across these repetitions.

*Definition 4.1 (Safety Violation).* To rigorously assess safety, we need to identify the safety-critical behavior conduct by the ego vehicle. Safety-critical behavior refers to system actions or states that can directly lead to a safety violation. We define *safety violations* within the test scenarios according to the following criteria:

- **I (Collision)**: The ego vehicle is involved in a collision with another vehicle.
- **II (Lane Departure)**: The ego vehicle crosses a solid lane marking.
- **III (Red Light Violation)**: The ego vehicle proceeds through a red traffic light.
- **IV (Motionless)**: The ego vehicle remains stationary for more than 15 seconds, despite the existence of a feasible path along its route.

### 4.3 Experimental Design

*4.3.1 Generalizability (RQ1).* We conducted experiments in the CARLA simulator across multiple scenarios and ADSs, integrating two online components. We conducted experiments across various maps of CARLA, each specifically selected to represent distinct driving scenarios:

- **Town01 (compact urban / dense traffic rules)**: A compact urban environment characterized by single-lane streets, tight intersections, and active traffic lights. This map is well-suited for assessing low-speed maneuvers in dense traffic and interactions among heterogeneous dynamic objects.
- **Town04 (highway-dominant / high-speed dynamics)**: A complex highway scenario featuring multiple lanes, overpasses, and merging lanes, making it ideal for evaluating high-speed driving and lane-changing behaviors.
- **Town07 (rural / navigation complexity)**: A large rural environment with varied road layouts, including unmarked roads without lane markings, and diverse route options. It features a richer road topology and higher navigation complexity, making it suitable for evaluating general driving policies, long-horizon navigation robustness, and interactions among heterogeneous dynamic objects.
- **Town10 (metropolitan / complex intersections + roundabouts)**: A metropolitan setting with dual-lane road layouts, roundabouts, and intersections, providing a challenging environment for testing urban driving policies and interactions among heterogeneous dynamic objects. As with Town04, traffic lights are not implemented in the Apollo version.

We evaluated three advanced ADSs within CARLA:

- **Apollo 8.0** [3]: Apollo represents a widely used, production-oriented autonomy stack with a full pipeline (localization/perception/planning/control). Evaluating on Apollo increases realism relative to CARLA's built-in controllers because failures may emerge from *multi-module interactions* (e.g., planning–control coupling) that do not exist in rule-based baselines.
- **Autoware (0.49.0)** [21]: Autoware is a major open-source ADS in the robotics/autonomous driving ecosystem. Including Autoware increases representativeness by covering a *different architecture and planning/control design* from Apollo, and by reflecting a widely used platform for academic and industry prototyping.
- **Traffic Manager** [18]: Traffic Manager provides a stable and reproducible baseline with deterministic behavior and minimal stack complexity. We include it to (i) separate improvements due to our method from failures attributable to complex perception/localization stacks, and (ii) quantify performance relative to a commonly used simulator-default ADS in prior CARLA-based testing work.

For the reinforcement learning (RL) component, we adapted the RL module from GARL [35], employing the DQN algorithm [40]. The state representation for each dynamic object consists of its relative position and orientation with respect to the ego vehicle, consistent with the $x$ defined in 3.3.3. To accommodate the heterogeneous nature of dynamic objects, we defined separate action spaces: one for vehicles and bicycles, and another for pedestrians. The action space for vehicles and bicycles includes seven discrete actions: **lane keeping**, **accelerate**, **brake**, **right-lane change with acceleration**, **right-lane change with deceleration**, **left-lane change with acceleration**, and **left-lane change with deceleration**. For pedestrians, the action space comprises four actions:

**forward**, **backward**, **left**, and **right**. The reward function is defined as the negative distance between the dynamic object and the ego vehicle, thereby incentivizing the agent to approach the ego vehicle. Each RL agent was pre-trained for 1,000 episodes on each map.

The generalizability of PTOP is quantitatively assessed using the **Safety Violation Rate**, which measures the proportion of test cases in which the ego vehicle experiences a safety violation (as we defined in Definition 4.1) during testing. Note that we do not include cross-map evaluation. This is because our approach does not involve any pre-training process; instead, all components are trained online during the testing phase.

*4.3.2 Effectiveness (RQ2).* In RQ2, we evaluate PTOP against three state-of-the-art (SOTA) baselines on the same CARLA maps used in RQ1, focusing on the most robust ADS identified in the RQ1 results. The selected SOTA baselines—*MOSAT* [46], *GARL* [35], and *KING* [23]—cover both online and testing with offline seed generation techniques.

- ***MOSAT*** [46]: We implement the MOSAT baseline using the NSGA-II algorithm [15]. The chromosome representation is consistent with that of PTOP (see §3.2.1), except that the *Dynamics Gene* encodes a discrete action sequence for each dynamic object. Each action sequence comprises 10 actions, selected from the same action space as the RL agent described in RQ1. Variation operators are applied as follows: upon mutation (according to a predefined rate), the chromosome representation is resampled; upon crossover, a single-point crossover is performed between the *Dynamics Gene* segments of two chromosomes. The multi-objective fitness function optimizes for: (1) *Safety Violation*, a binary indicator denoting whether a test case induces a safety violation; and (2) *Diversity*, which quantifies the uniqueness of each chromosome as the average Euclidean distance to all other chromosomes in the generation. The diversity metric is computed as [30, 31, 35]:

$$D_i = \frac{1}{k} \sum_{j=1}^{k} d\left(x_i, x_j\right)$$

  where $D_i$ denotes the diversity score for the $i$-th chromosome, $x_i$ and $x_j$ are chromosome representations, $d(\cdot, \cdot)$ is the Euclidean distance, and $k$ is the generation size. We reproduce the action patterns and trigger conditions as specified in MOSAT [46].
- ***GARL*** [35]: We adapt GARL from its original UAV marker-based landing context. In this baseline, multiple single-agent RL models independently control the surrounding vehicles, without inter-agent communication, within scenarios generated by a genetic algorithm. For the GA component, we employ NSGA-II [15], utilizing the same chromosome representation, variation operators, and multi-objective fitness function as in the MOSAT baseline. The RL component is implemented identically to that described in RQ1.
- ***KING*** [23]: We implement the online testing approach as detailed in 3.4, with each scenario initialized randomly.

We use following metrics to assess this RQ:

- **Safety Violation Rate**: As defined in RQ1, this metric quantifies the proportion of test cases in which the ego vehicle experiences a safety violation.
- **TOP-10**: This metric measures the number of test rounds required to identify the first ten safety violations, thereby reflecting the efficiency of each method in uncovering critical failures.
- **Parameter Distance**: Inspired by novelty search [30, 31], this metric quantifies the diversity among the generated safety violations. A higher value indicates greater diversity. The metric

is formally defined as:

$$\rho(x) = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{1}{n}\sum_{j=1}^{n} d(x_i, x_j)\right]$$

where $x_i$ and $x_j$ represent the scenario representation vectors of the $i$-th and $j$-th safety-violating cases, respectively. Each vector encodes the positions and orientations of the ego vehicle and dynamic objects. The function $d(x_i, x_j)$ denotes the Euclidean distance, and $n$ is the total number of generated safety violations. All gene values are normalized to the range $[0, 1]$. Notably, the maximum possible distance between two chromosome representations is 1.

- **Map Coverage**: This metric assesses the fraction of map spawn points utilized by dynamic objects near the ego vehicle in safety-violating scenarios. For each safety violation, we collect the spawn positions of dynamic objects within a radius of $R = 50$ m from the ego vehicle, map each position to its nearest spawn point within a threshold of 0.05 m, and remove duplicates. Let $\mathcal{S}_m$ denote the set of all map spawn points and $\mathcal{U}_m \subseteq \mathcal{S}_m$ the subset matched by the above procedure. The map coverage is then computed as:

$$\text{MapCoverage} = \frac{|\mathcal{U}_m|}{|\mathcal{S}_m|} \times 100\%.$$

- **Trajectory Coverage**: To quantify the behavioral diversity of dynamic objects, we use *trajectory coverage*. Since the online component involves both discrete and continuous action spaces, it is difficult to measure diversity directly in the action space (e.g., via distance-based metrics). Following [35], we treat trajectories as the outcome of behaviors and compute coverage in a map-aware manner. Concretely, we collect all positions of each dynamic object during a run, map each position to its nearest waypoint within a threshold of 0.05 m, and remove duplicates. Let $\mathcal{S}_t$ denote the set of all map waypoints and $\mathcal{U}_t \subseteq \mathcal{S}_t$ the subset matched by the above procedure. The trajectory coverage rate is computed as:

$$\text{TrajectoryCoverage} = \frac{|\mathcal{U}_t|}{|\mathcal{S}_t|} \times 100\%.$$

- **Time Consumption**: This metric records the total time required by each method to exhaust its allotted test budget. While accuracy is paramount, excessive test case generation time can impede practical adoption in real-world testing pipelines.

Additionally, we compare our method with TARGET [16]. Since TARGET follows a different testing methodology, a direct comparison under the same set of metrics is not applicable. Instead, we compare the two methods in terms of *efficiency* in finding safety violations.

To ensure a fair comparison, we standardize the safety-violation definitions and focus on the overlap between the two methods: *collision*, *lane departure*, and *motionless*. We perform this comparison on all maps used in RQ1 and report the overall result, with both Apollo and Autoware as the system under test. As an efficiency measure, we report the average number of violations of each type found within a fixed time budget of 6 hours (4 runs).

*4.3.3 Ablation Study (RQ3).* To evaluate the contributions of individual components in PtoP, we conduct an ablation study with two settings:

- ***PtoP with vs. without ARSG***: ARSG is replaced with a random seed generator to assess its role in discovering additional failure modes exploitable by SVGD.
- ***PtoP with SVGD vs. with GA***: SVGD and ART are replaced with a GA-based generator (same as MOSAT in RQ2) to compare ARSG+SVGD against GA in producing diverse, failure-inducing scenarios.

Table 1. Averaged ADS safety violation rate (%) and across different CARLA maps when PtoP using KING as online part.

| Map | Metric | Apollo | Autoware | Traffic Manager |
|---|---|---|---|---|
| **Town1** | Violation Rate (%) | **23.06** | 28.68 | 28.64 |
| **Town4** | Violation Rate (%) | **18.31** | 22.17 | 21.47 |
| **Town7** | Violation Rate (%) | **30.13** | 37.18 | 37.85 |
| **Town10** | Violation Rate (%) | **19.81** | 27.58 | 29.45 |

Table 2. Averaged ADS safety violation rate (%) across different CARLA maps when PtoP using RL as online part.

| Map | Metric | Apollo | Autoware | Traffic Manager |
|---|---|---|---|---|
| **Town1** | Violation Rate (%) | **20.94** | 31.13 | 33.74 |
| **Town4** | Violation Rate (%) | **18.75** | 24.52 | 23.35 |
| **Town7** | Violation Rate (%) | **28.00** | 35.31 | 34.75 |
| **Town10** | Violation Rate (%) | **21.31** | 31.03 | 33.45 |

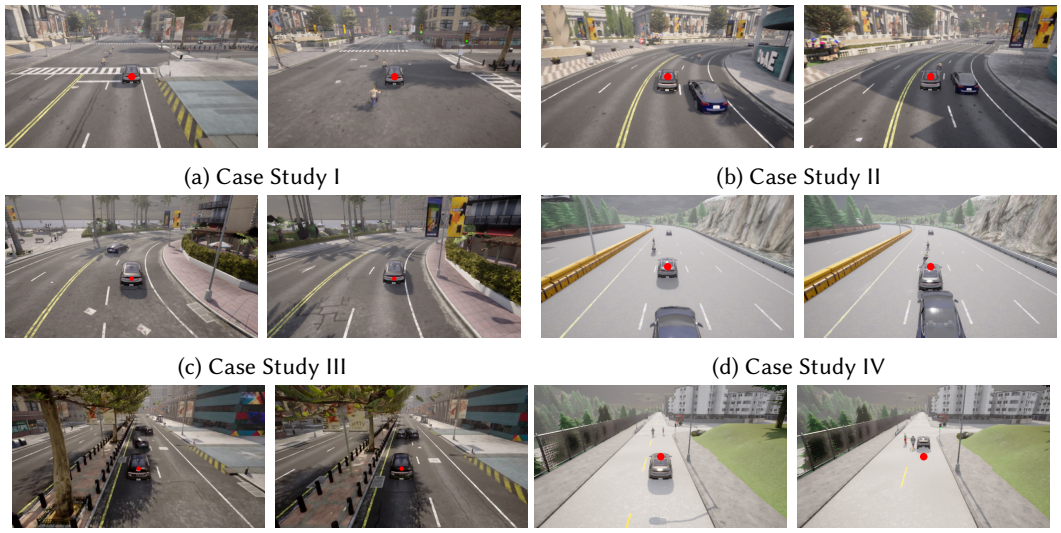
All experiments use the same maps, ADS configurations, and metrics as RQ2 for fair comparison.

*4.3.4 Fidelity (RQ4).* To validate the realism of the generated safety violations, we conducted a user study involving experienced drivers. We randomly selected 12 videos from our record and recruited 30 experienced drivers globally through Prolific [42]. Participants, aged between 18 and 60 years and holding valid driver's licenses, represented diverse genders. Each participant evaluated the realism of surrounding vehicle trajectories by answering the question: *"How realistic are the surrounding vehicles' trajectories in this video?"* Responses were recorded using a Likert scale with the following categories: *"Very unrealistic," "Slightly unrealistic," "Neutral," "Slightly realistic,"* and *"Very realistic."* If a participant rated a video below *"Neutral,"* they were asked to provide a reason for their rating, the survey can be found **here** (enter any worker ID). Additionally, we calculated Weighted Fleiss' Kappa [16] to measure inter-rater agreement among participants. Detailed demographic information can be found **here**. The user study interface and the Weighted Fleiss' Kappa calculation are provided in **Sections 1 and 2 of the supplementary material**.

### 4.4 Generalizability of PtoP (RQ1)

Tables 1 and 2 summarize the generalizability of PtoP in CARLA across Town 1, Town 4, Town 7, and Town 10. We evaluate PtoP with Apollo, Autoware, and Traffic Manager under different online methods. Overall, both online methods integrate well with PtoP and can uncover safety violations across all scenarios and ADSs. Online RL performs slightly better than KING in Town 4 and Town 10; however, KING outperforms RL in Town 1 and Town 7. This is likely because Town 1 and Town 7 are more rural, where most roads are single-lane; with RL's discrete action space, lane-change actions are limited in these maps. In contrast, KING uses a gradient-based strategy to synthesize continuous actions, which gives it an advantage in Town 1 and Town 7.

PtoP consistently records fewer safety violations with Apollo, whereas Traffic Manager and Autoware exhibit higher and comparable violation rates. This is likely because Traffic Manager implements only basic autonomous-driving functions (e.g., stopping for vehicles ahead). In addition, Autoware may fail to detect static or slow-moving vehicles, which can cause the ego vehicle to collide with the vehicle in front; this behavior is consistent with a previously reported Autoware–CARLA issue [16]. Additionally, all ADS configurations show the fewest safety violations in Town 4,



(a) Case Study I (b) Case Study II

(c) Case Study III (d) Case Study IV

(e) Case Study V (f) Case Study VI

Fig. 2. Case study.

likely due to its significantly larger area, which reduces the frequency of interactions that lead to safety violations.

We conducted a case study to manually analyze the causes of selected safety-violation cases identified by PTOP in Apollo. We selected cases that involve heterogeneous dynamic objects and exhibit diverse causes of safety violations. **The vehicle annotated with a red dot in the image is the ego vehicle.**

*4.4.1 Case Study I.* The ego vehicle collides with multiple cyclists due to unexpected lane changes by surrounding cyclists. In Figure 2 (a), the ego vehicle and several cyclists are passing through an intersection when one cyclist accelerates to merge into the ego vehicle's lane, triggering a collision.

*4.4.2 Case Study II.* The ego vehicle collides with the cyclist due to acceleration of a following vehicle. As shown in Figure 2 (b), on a two-lane road, the ego vehicle is traveling in the left lane, a cyclist is ahead in the right lane, and another vehicle is driving behind the cyclist. When the following vehicle accelerates, the cyclist gives way by merging into the left lane (the ego vehicle's lane). Due to the short headway, the ego vehicle collides with the cyclist.

*4.4.3 Case Study III.* Figure 2 (c) illustrates a scenario where the ego vehicle crosses a solid line on the road. This occurs when a surrounding vehicle attempts to merge into the ego vehicle's lane while accelerating. Due to the close proximity between the ego vehicle and the surrounding vehicle, the ego vehicle attempts to avoid a collision by steering slightly to the right, inadvertently crossing the solid line.

*4.4.4 Case Study IV:.* Figure 2(d) illustrates a scenario where the ego vehicle hits a cyclist. On a four-lane road, the ego vehicle is traveling with another vehicle following behind. A cyclist suddenly merges into the ego vehicle's lane; the ego vehicle brakes sharply but still collides with the cyclist. Immediately after, the following vehicle rear-ends the ego vehicle.

Table 3. Baseline results across towns. Best results are in bold.

| Method | Town | Safety Violation % | Top-10 | Parameter distance | Map Coverage % | Trajectory Coverage % | Time Consumption (hours) |
|---|---|---|---|---|---|---|---|
| *PrioP* | Town1 | **23.06**% | **39.5** | **0.331** | **31.17**% | 51.4% | 6 |
| | Town4 | **18.31**% | **62.5** | **0.317** | **20.53**% | 61.4% | 6 |
| | Town7 | **30.13**% | **42** | **0.319** | **79.65**% | 90.3% | 6 |
| | Town10 | **19.81**% | **45.25** | **0.315** | **63.54**% | 89.2% | 6 |
| *MOSAT [46]* | Town1 | 19.38% | 49.5 | 0.302 | 25.96% | 38.6% | 6 |
| | Town4 | 15.69% | 70.5 | 0.292 | 15.31% | 40.6% | 6 |
| | Town7 | 24.00% | 60.5 | 0.281 | 55.70% | 69.2% | 6 |
| | Town10 | 17.38% | 52.75 | 0.291 | 57.04% | 70.3% | 6 |
| *GARL [35]* | Town1 | 18.06% | 52 | 0.303 | 25.22% | 48.6% | 6 |
| | Town4 | 15.00% | 73 | 0.292 | 15.78% | 55.3% | 6 |
| | Town7 | 24.63% | 71.25 | 0.287 | 56.48% | 80.6% | 6 |
| | Town10 | 17.89% | 58.75 | 0.288 | 54.41% | 85.6% | 6 |
| *KING [23]* | Town1 | 21.00% | 45 | 0.302 | 25.25% | 53.7% | 6 |
| | Town4 | 15.19% | 73 | 0.292 | 15.23% | 62.4% | 6 |
| | Town7 | 26.13% | 53.25 | 0.291 | 75.18% | 91.5% | 6 |
| | Town10 | 17.13% | 56.50 | 0.292 | 56.31% | 90.6% | 6 |

Table 4. Two-sided paired $t$-test results of baselines vs. PrioP (n=4 runs): $p$-values, 95% confidence intervals (CIs), and effect sizes (Cohen's $d$ for paired samples)

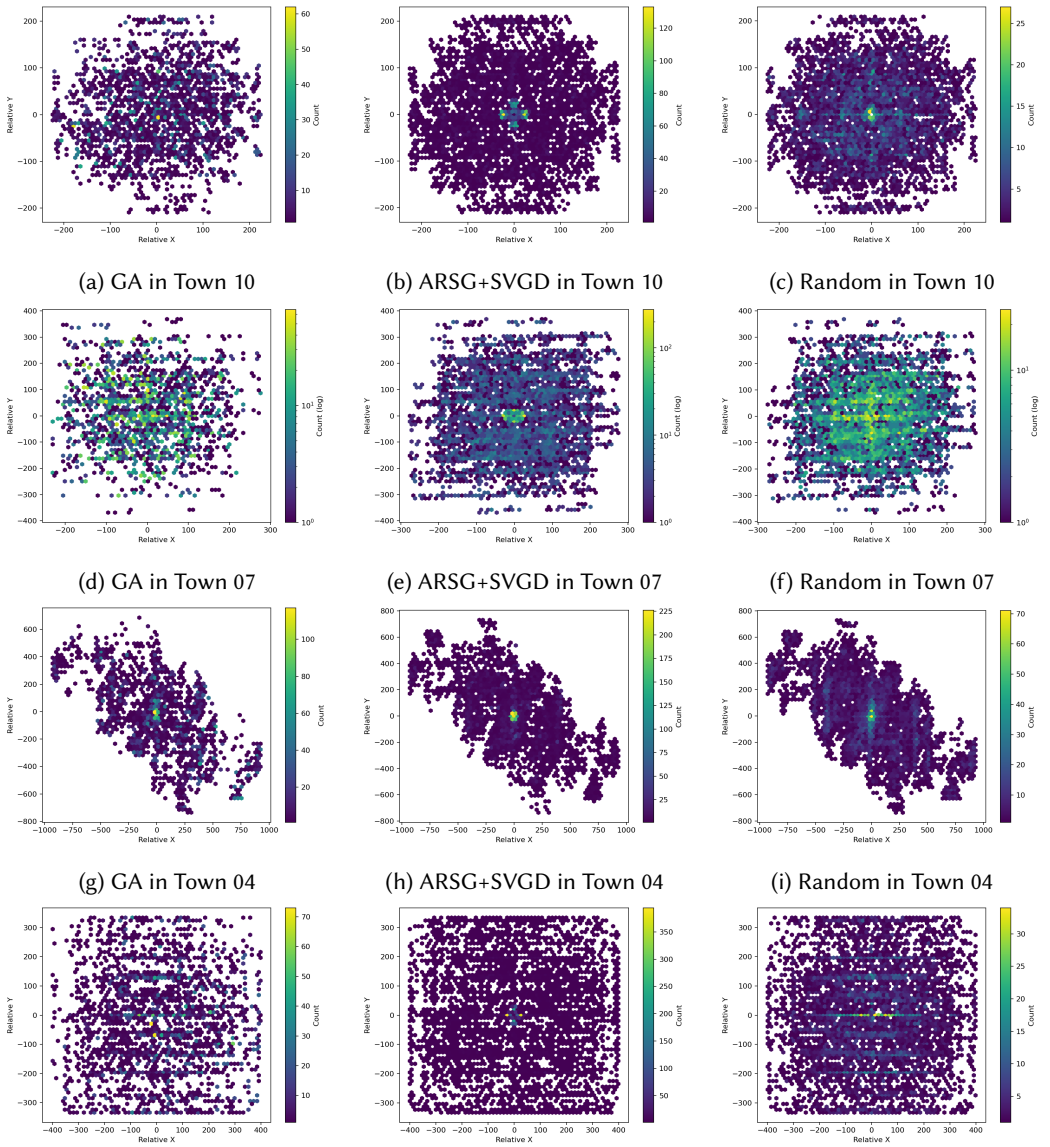
| Method | Metric | Town1 | | | Town4 | | | Town7 | | | TOWN10 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $p$-values | 95% CI | Effect | $p$-values | 95% CI | Effect | $p$-values | 95% CI | Effect | $p$-values | 95% CI | Effect |
| MOSAT | Violation Rate | 0.0222 | [0.0100, 0.0637] | 2.185 | 0.0363 | [0.0032, 0.0493] | 1.810 | 0.0175 | [0.0204, 0.1021] | 2.383 | 0.1751 | [−0.0195, 0.0682] | 0.884 |
| | TOP-K | 0.0103 | [−15.51, −4.49] | -2.887 | 0.0902 | [−18.31, 2.31] | -1.234 | 0.1163 | [−45.39, 8.39] | -1.095 | 0.0886 | [−17.09, 2.09] | -1.244 |
| | Parameter Distance | 0.0000 | [0.0219, 0.0306] | 13.688 | 0.0003 | [0.0251, 0.0390] | 6.044 | 0.0108 | [0.0116, 0.0484] | 2.781 | 0.0172 | [0.0075, 0.0350] | 2.442 |
| | Map Coverage | 0.0002 | [3.21, 6.39] | 6.383 | 0.0009 | [2.06, 3.74] | 5.306 | 0.0119 | [1.66, 9.64] | 2.719 | 0.0169 | [0.85, 4.88] | 2.462 |
| GARL | Violation Rate | 0.0046 | [0.0331, 0.0669] | 4.141 | 0.0116 | [0.0134, 0.0454] | 2.890 | 0.0295 | [0.0086, 0.0826] | 2.067 | 0.3211 | [−0.0186, 0.0551] | 0.585 |
| | TOP-K | 0.0171 | [−18.01, −1.99] | -2.429 | 0.0187 | [−18.76, −2.74] | -2.374 | 0.0359 | [−45.06, −2.94] | -1.905 | 0.0076 | [−23.52, −7.48] | -3.347 |
| | Parameter Distance | 0.0002 | [0.0230, 0.0330] | 8.686 | 0.0001 | [0.0292, 0.0413] | 8.043 | 0.0120 | [0.0097, 0.0418] | 2.673 | 0.0306 | [0.0035, 0.0355] | 2.055 |
| | Map Coverage | 0.0004 | [4.03, 7.87] | 5.908 | 0.0084 | [0.47, 2.82] | 3.050 | 0.0001 | [8.28, 12.74] | 9.252 | 0.0015 | [2.48, 6.11] | 4.945 |
| KING | Violation Rate | 0.2793 | [−0.0233, 0.0095] | -0.622 | 0.0040 | [0.0223, 0.0502] | 3.509 | 0.1010 | [−0.0154, 0.0936] | 1.140 | 0.0349 | [−0.0019, 0.0669] | 1.669 |
| | TOP-K | 0.0724 | [−12.82, 1.82] | -1.466 | 0.0031 | [−19.20, −5.30] | -3.736 | 0.4248 | [−22.59, 10.59] | -0.461 | 0.0114 | [−21.85, −4.65] | -2.778 |
| | Parameter Distance | 0.0000 | [0.0270, 0.0370] | 14.000 | 0.0004 | [0.0230, 0.0370] | 5.886 | 0.0106 | [0.0116, 0.0504] | 2.809 | 0.0225 | [0.0050, 0.0438] | 2.257 |
| | Map Coverage | 0.0008 | [3.61, 5.40] | 5.158 | 0.0028 | [1.39, 3.39] | 3.888 | 0.1351 | [−1.33, 9.34] | 0.960 | 0.0268 | [0.32, 5.64] | 1.910 |

*4.4.5 Case Study V:.* Figure 2 (e) illustrates a scenario where the ego vehicle experiences a motionless state despite having a feasible route. Initially, two surrounding vehicles collide while attempting a lane change and come to a stop on the road. Later, a third surrounding vehicle attempts to overtake the ego vehicle by changing lanes with acceleration to find a new route. However, due to the obstructed view caused by the ego vehicle and the stationary collided vehicles ahead, the third vehicle collides with the already-stopped vehicles after completing its lane change. At this point, the ego vehicle has the option to take the right lane to continue its route but remains stuck on the road, ultimately leading to a motionless.

*4.4.6 Case Study VI:.* Figure 2 (f) illustrates a near miss with pedestrians. At the beginning, the ego vehicle is waiting at a red light while two pedestrians are crossing the road. When the light turns green, the ego vehicle starts moving; however, the pedestrian attempts to cross the road diagonally, and the ego vehicle does not slow down, accelerating toward the pedestrian—creating a highly dangerous situation.

## 4.5 Effectiveness (RQ2)

Table 3 reports quantitative results comparing PrioP with baseline methods across multiple Apollo maps. Overall, PrioP consistently outperforms all baselines in **Safety Violation Rate** (by up to **27.68%**), **TOP-10** (by up to **24.04%**), **parameter distance** (by up to **9.60%**), and **map coverage** (by up to **16.78%**), while incurring no additional runtime overhead. Regarding trajectory coverage, performance mainly depends on the online control component. Due to the gradient-based controller used in KING and its continuous action space, exploration tends to be broader than that of RL-based and GA-based control methods. We then conduct significant tests on the experimental results.




(a) GA in Town 10 (b) ARSG+SVGD in Town 10 (c) Random in Town 10

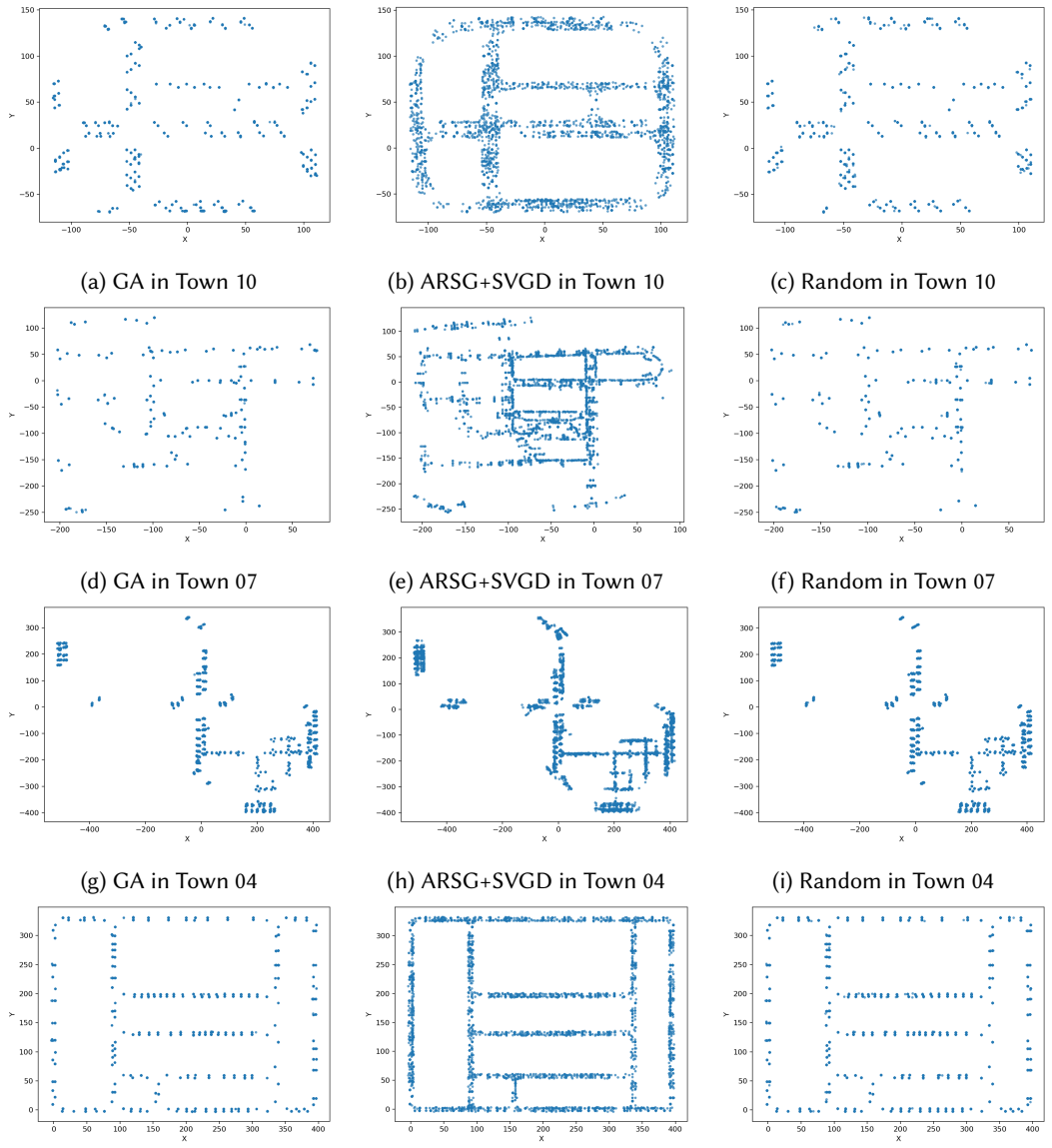
(d) GA in Town 07 (e) ARSG+SVGD in Town 07 (f) Random in Town 07

(g) GA in Town 04 (h) ARSG+SVGD in Town 04 (i) Random in Town 04

(j) GA in Town 01 (k) ARSG+SVGD in Town 01 (l) Random in Town 01

Fig. 3. Scatter plot of the initial relative position between ego vehicle and dynamic objects across different maps generated by different offline methods.

From Table 4, we observe that PtoP differs significantly from other methods on most metrics, as indicated by the paired two-sided $t$-tests. In particular, for **parameter distance** and **map coverage**—the strengths of PtoP —PtoP consistently achieves statistically significant improvements over all baselines, with 95% confidence intervals largely excluding zero and generally large effect sizes.



(a) GA in Town 10 (b) ARSG+SVGD in Town 10 (c) Random in Town 10

(d) GA in Town 07 (e) ARSG+SVGD in Town 07 (f) Random in Town 07

(g) GA in Town 04 (h) ARSG+SVGD in Town 04 (i) Random in Town 04

(j) GA in Town 01 (k) ARSG+SVGD in Town 01 (l) Random in Town 01

Fig. 4. Scatter plot of the absolute initial position of dynamic objects across different maps generated by different offline methods.

To validate the assumptions of the $t$-test, we conducted the Shapiro–Wilk test on the paired differences, which showed no significant deviations from normality (all $p > 0.05$). Given the small sample size ($n = 4$), we further applied the non-parametric Wilcoxon signed-rank test. While the Wilcoxon test does not always reach statistical significance (e.g., $p = 0.125$ in several cases), this is expected due to its limited statistical power with small samples. Importantly, the observed

Table 5. Ablation study across towns.

| Method | Town | Safety Violation % | Top-10 | Parameter distance | Map Coverage % | Time Consumption (hours) |
|---|---|---|---|---|---|---|
| *PtoP without ART* | Town1 | 20.21% | 50.25 | 0.301 | 29.70% | 6 |
| | Town4 | 15.06% | 75.75 | 0.292 | 20.67% | 6 |
| | Town7 | 27.19% | 51 | 0.297 | 70.25% | 6 |
| | Town10 | 16.25% | 50.75 | 0.292 | 65.18% | 6 |
| *PtoP with GA* | Town1 | 20.69% | 49.25 | 0.3 | 25.51% | 6 |
| | Town4 | 14.38% | 74.5 | 0.291 | 15.51% | 6 |
| | Town7 | 27.25% | 51 | 0.283 | 59.15% | 6 |
| | Town10 | 18.51% | 56 | 0.293 | 56.49% | 6 |

performance trends remain consistent with those of the paired $t$-test, and the effect sizes remain large.

This superior performance stems from the diverse, failure-inducing initial scenarios generated by our approach, enabling broader parameter space exploration and improved coverage; additional cross-town and cross-agent significance tests are reported in **Tables 1–2 in the supplementary material**. To substantiate this, Figures 3 and 4 show (i) scatter plots of initial *relative* positions between the ego vehicle and dynamic objects across maps generated by different offline methods, and (ii) scatter plots of the *absolute* initial positions of dynamic objects.

For the relative position, which serves as a proxy for efficiency, the distribution is expected to be approximately normal: intuitively, dynamic objects nearer to the ego vehicle have a higher likelihood of causing a safety violation. From Figure 3, we observe that ARSG+SVGD concentrates samples near the peak of this distribution, whereas the other baselines are more evenly spread; GA, in particular, clusters around distinct local minima, while Random remains broadly uniform. This behavior arises because SVGD combines an attractive term that pulls particles toward high-hazard modes. In contrast, GA tends to discover multiple local minima and, through its inheritance mechanism, propagates these minima to subsequent generations, which limits efficiency.

For the absolute position, which serves as a proxy for diversity, the samples should be broadly and evenly distributed across the map. Figure 4 shows that ART+SVGD places dynamic objects more evenly and in a denser, more uniform pattern across the map, whereas GA and Random are noticeably sparser. Together with the numerical results, these observations indicate that PtoP produces more diverse and failure-inducing initial scenarios.

Moreover, regarding the number of safety violations found within 6 hours, although the methodologies differ, under the same evaluation settings our method achieves roughly twice the safety-violation discovery rate of TARGET (91.31 vs. 40.5). This result highlights the importance of search-based testing.

PtoP is not highly sensitive to SVGD parameters within reasonable ranges. We use 5 particles in all experiments as it provides a good balance between exploration and exploitation; increasing to 20 particles in Town4 reduces parameter distance to 0.25, reflecting weaker exploration under online training. The kernel bandwidth is adaptively computed using the median heuristic, making it self-adjusting across towns and episodes. Temperature is set to 1 by default; lower values (e.g., 0.1) still yield stable behavior ( 0.29 distance), whereas very high temperatures (e.g., 5) collapse particles into a single mode and cause unrealistic stacking. Overall, across all reasonable settings, PtoP retains its performance gains. Moreover, we further analyze the failure taxonomy in **the Figure 2 of supplementary material**.

### 4.6 Ablation Study (RQ3)

Table 5 presents the ablation study results. For PtoP with GA, we reach the same conclusion as in RQ2 i.e. GA exhibits limited performance in **map coverage** and **diversity**, which is aligned with

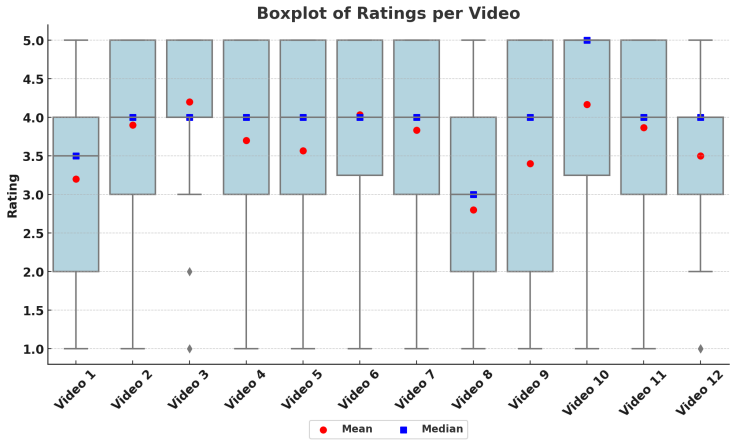



Fig. 5. Boxplots for user ratings on sampled video with outliers.

(a) Example of Video 3 scenario

(b) Example of Video 12 scenario

Fig. 6. Outlier analysis.

MOSAT and GARL. For PTOP without ART, the method still maintains high **map coverage**, but the **parameter distance** drops to the level of GA and Random. This aligns with our hypothesis: ART tends to identify additional failure modes, while SVGD explores within each discovered mode. Consequently, removing ART reduces PTOP's ability to uncover more failure modes, yet its exploration capability remains, thereby preserving coverage.

### 4.7 Fidelity (RQ4)

Figure 5 presents a box plot of realism ratings from the user study, where the red and blue dots represent the mean and median ratings, respectively. The median rating for all videos exceeds 3, indicating a "Neutral" or higher realism perception. Only video 8 has a mean rating below 3. The weighted Fleiss' Kappa for user ratings is **0.714**, indicating "Substantial Agreement." However, outliers remain in videos 3 and 12, analyzed in the next section.

*4.7.1 Video 3 & Video 12 Outlier Analysis.* In Video 3, we observe two outliers. One comment states, "I don't think this would happen in the real world". This likely refers to the left image of Figure 6, where two surrounding vehicles collide due to concurrent lane changes. In Video 12, one outlier comment is "They were constantly changing lanes". The right image of Figure 6 shows vehicles attempting multiple lane changes within a short period. Currently, our framework relies

on existing online methods; future work will develop stronger online components and incorporate additional regulations (e.g., rule-based constraints) to improve realism.

## 5 Threats to Validity

**Internal Validity.** A key threat lies in the choice of online testing methods. RL-based testers [35] suffer from *action realism* issues, where generated behaviors may diverge from plausible driving maneuvers, while gradient-based testers [23] face the *catch-on-collision* problem, where optimization halts once a collision occurs. These limitations may affect the extent to which discovered violations reflect realistic driving scenarios. Nevertheless, our framework is *plug-and-play* and integrates these state-of-the-art online testing methods to ensure a fair comparison; developing more robust online testers is orthogonal and left as future work.

**Construct Validity.** The validity of our evaluation also depends on the chosen baselines. To reduce bias, we emphasize the complementary roles of the offline seed generator and the online tester through a hazard learning model, and we compare against both recent offline generators and leading online testers. This setup increases confidence that performance gains arise from our design rather than from limited baseline coverage.

**External Validity.** Although CARLA-based simulation cannot fully replicate real-world sensing noise and long-tail edge cases, our design mitigates the external validity by demonstrating that observed effects persist across diverse road structures and autonomy architectures rather than emerging from a single simulator configuration or system implementation. Concretely, the selected CARLA maps span compact urban (Town1), metropolitan multi-intersection (Town10), highway-dominant (Town4), and rural navigation-complex (Town7) scenarios, thereby covering heterogeneous road topologies, traffic regulations, speed regimes, and interaction patterns. This reduces the risk that results are tied to a single driving context. In parallel, we evaluate three architecturally distinct ADSs—Apollo 8.0 and Autoware as full-stack, industry-scale open-source systems with different planning and control designs, and Traffic Manager as a deterministic, low-complexity baseline—thereby limiting conclusions to neither a specific autonomy stack nor a particular perception–planning–control coupling. ISO 34502 and ISO 21448 (SOTIF) are not threats but complementary contexts: our tool offers *exploratory* stress testing for early-stage violation discovery, aligned with SOTIF's focus on performance limitations under nominal conditions.

## 6 Conclusion

We presented PtoP, a plug-and-play framework that couples an SVGD-based offline seed generator with online testers via a hazard model, enabling diverse and realistic failure-inducing scenarios. Experiments in CARLA with Apollo 8.0 and Traffic Manager show that PtoP outperforms state-of-the-art baselines, improving violation rates by up to 27.68%, diversity by 9.6%, and map coverage by 16.78%, with scenarios rated realistic by human judges. PtoP 's extensible design allows seamless integration of future online testers and hazard models. A key direction is to enhance both offline seeding and online testing to support higher-fidelity simulation environments, thereby narrowing the simulation–reality gap and enabling safer, more robust ADS deployment.

## Data-Availability Statement

Our artifact is available at https://doi.org/10.5281/zenodo.19625701 [34] (mirror: https://github.com/lfeng0722/PtoP).

## Acknowledgments

This work is supported by the Australian Research Council grants FT240100269 and DP210102447.

## References


[1] [n. d.]. Baidu Apollo team (2017), Apollo: Open Source Autonomous Driving, howpublished = https://github.com/ApolloAuto/apollo, note = Accessed: 2019-02-11.

[2] Raja Ben Abdessalem, Annibale Panichella, Shiva Nejati, Lionel C Briand, and Thomas Stifter. 2018. Testing autonomous cars for feature interaction failures using many-objective search. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 143–154.

[3] ApolloAuto. 2024. Apollo. https://github.com/ApolloAuto/apollo.

[4] Robyn G Attewell and Stephen Ginpil. 1994. *Crashes resulting in car occupant fatalities: Frontal impacts*. Number CR 138. Australian Government Pub. Service.

[5] Australian Government Department of Infrastructure, Transport, Regional Development, Communications and the Arts. 2025. Monthly Road Deaths - Road Safety Data Hub. https://datahub.roadsafety.gov.au/progress-reporting/monthly-road-deaths Accessed: 2025-01-27.

[6] Earl T Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. 2014. The oracle problem in software testing: A survey. *IEEE transactions on software engineering* 41, 5 (2014), 507–525.

[7] Raja Ben Abdessalem, Shiva Nejati, Lionel C Briand, and Thomas Stifter. 2016. Testing advanced driver assistance systems using multi-objective search and neural networks. In *Proceedings of the 31st IEEE/ACM international conference on automated software engineering*. 63–74.

[8] Michele Bertoncello and Dominik Wee. 2015. Ten ways autonomous driving could redefine the automotive world. *McKinsey & Company* 6 (2015).

[9] Lukas Birkemeyer, Tobias Pett, Andreas Vogelsang, Christoph Seidl, and Ina Schaefer. 2022. Feature-Interaction Sampling for Scenario-based Testing of Advanced Driver Assistance Systems. In *Proceedings of the 16th International Working Conference on Variability Modelling of Software-Intensive Systems*. 1–10.

[10] Li Chen, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li. 2024. End-to-end autonomous driving: Challenges and frontiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024).

[11] Tsong Yueh Chen, Hing Leung, and Ieng Kei Mak. 2004. Adaptive random testing. In *Annual Asian Computing Science Conference*. Springer, 320–329.

[12] Yuntianyi Chen, Yuqi Huai, Shilong Li, Changnam Hong, and Joshua Garcia. 2024. Misconfiguration Software Testing for Failure Emergence in Autonomous Driving Systems. *Proceedings of the ACM on Software Engineering* 1, FSE (2024), 1913–1936.

[13] Mingfei Cheng, Yuan Zhou, Xiaofei Xie, Junjie Wang, Guozhu Meng, and Kairui Yang. 2024. Decictor: Towards Evaluating the Robustness of Decision-Making in Autonomous Driving Systems. *arXiv preprint arXiv:2402.18393* (2024).

[14] Erwin De Gelder and Jan-Pieter Paardekooper. 2017. Assessment of automated driving systems using real-life scenarios. In *2017 ieee intelligent vehicles symposium (iv)*. IEEE, 589–594.

[15] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and Tanaka Meyarivan. 2000. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Parallel Problem Solving from Nature PPSN VI: 6th International Conference Paris, France, September 18–20, 2000 Proceedings 6*. Springer, 849–858.

[16] Yao Deng, Jiaohong Yao, Zhi Tu, Xi Zheng, Mengshi Zhang, and Tianyi Zhang. 2023. Target: Traffic rule-based test generation for autonomous driving systems. *arXiv preprint arXiv:2305.06018* (2023).

[17] Yao Deng, Xi Zheng, Mengshi Zhang, Guannan Lou, and Tianyi Zhang. 2022. Scenario-based test reduction and prioritization for multi-module autonomous driving systems. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 82–93.

[18] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 2017. CARLA: An open urban driving simulator. In *Conference on robot learning*. PMLR, 1–16.

[19] Hamid Ebadi, Mahshid Helali Moghadam, et al. 2021. Efficient and effective generation of test cases for pedestrian detection-search-based software testing of Baidu Apollo in SVL. In *2021 IEEE International Conference on Artificial Intelligence Testing (AITest)*. IEEE, 103–110.

[20] Shuo Feng, Haowei Sun, Xintao Yan, et al. 2023. Dense reinforcement learning for safety validation of autonomous vehicles. *Nature* 615, 7953 (2023).

[21] Autoware Foundation. 2025. Autoware: Open-Source Software for Autonomous Driving. https://github.com/autowarefoundation/autoware. Accessed: 2025-02-18.

[22] Alessio Gambi, Tri Huynh, and Gordon Fraser. 2019. Generating effective test cases for self-driving cars from police reports. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 257–267.

[23] Niklas Hanselmann, Katrin Renz, Kashyap Chitta, Apratim Bhattacharyya, and Andreas Geiger. 2022. King: Generating safety-critical driving scenarios for robust imitation via kinematics gradients. In *European Conference on Computer Vision*. Springer, 335–352.

[24] Florian Hauer, Ilias Gerostathopoulos, Tabea Schmidt, and Alexander Pretschner. 2020. Clustering traffic scenarios using mental models as little as possible. In *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 1007–1012.
[25] Yuqi Huai, Sumaya Almanee, Yuntianyi Chen, Xiafa Wu, Qi Alfred Chen, and Joshua Garcia. 2023. sceno RITA: Generating Diverse, Fully-Mutable, Test Scenarios for Autonomous Vehicle Planning. *IEEE Transactions on Software Engineering* (2023).
[26] Yuqi Huai, Yuntianyi Chen, Sumaya Almanee, Tuan Ngo, Xiang Liao, Ziwen Wan, Qi Alfred Chen, and Joshua Garcia. 2023. Doppelgänger test generation for revealing bugs in autonomous driving software. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2591–2603.
[27] Florian Klück, Yihao Li, Mihai Nica, Jianbo Tao, and Franz Wotawa. 2018. Using ontologies for test suites generation for automated and autonomous driving functions. In *2018 IEEE International symposium on software reliability engineering workshops (ISSREW)*. IEEE, 118–123.
[28] Mark Koren, Saud Alsaif, Ritchie Lee, and Mykel J Kochenderfer. 2018. Adaptive stress testing for autonomous vehicles. In *2018 IEEE Intelligent Vehicles Symposium*. IEEE.
[29] Fred Lambert. 2016. Understanding the fatal tesla accident on autopilot and the nhtsa probe. *Electrek, July* 1 (2016), 1.
[30] Joel Lehman and Kenneth O Stanley. 2011. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation* 19, 2 (2011), 189–223.
[31] Joel Lehman, Kenneth O Stanley, et al. 2008. Exploiting open-endedness to solve problems through the search for novelty.. In *ALIFE*. 329–336.
[32] Guanpeng Li, Yiran Li, Saurabh Jha, et al. [n. d.]. Av-fuzzer: Finding safety violations in autonomous driving systems. In *2020 IEEE 31st international symposium on software reliability engineering (ISSRE)*.
[33] Pingfei Li, Xinyu Zhu, Yao Ren, Zhengping Tan, Wenhao Hu, You Zhang, and Chang Xu. 2024. Generalization of cut-in pre-crash scenarios for autonomous vehicles based on accident data. *Scientific reports* 14, 1 (2024), 17664.
[34] Linfeng Liang, Xiao Cheng, Tsong Yueh Chen, and Xi Zheng. 2025. Artifact for: From Particles to Perils: SVGD-Based Hazardous Scenario Generation for Autonomous Driving Systems Testing. https://doi.org/10.5281/zenodo.19625701
[35] Linfeng Liang, Yao Deng, Kye Morton, Valtteri Kallinen, Alice James, Avishkar Seth, Endrowednes Kuantama, Subhas Mukhopadhyay, Richard Han, and Xi Zheng. 2023. RLaGA: A Reinforcement Learning Augmented Genetic Algorithm For Searching Real and Diverse Marker-Based Landing Violations. *arXiv preprint arXiv:2310.07378* (2023).
[36] Qiang Liu and Dilin Wang. 2016. Stein variational gradient descent: A general purpose bayesian inference algorithm. *Advances in neural information processing systems* 29 (2016).
[37] Chengjie Lu, Yize Shi, et al. 2022. Learning configurations of operating environment of autonomous vehicles to maximize their collisions. *IEEE Transactions on Software Engineering* 49, 1 (2022), 384–402.
[38] Yuteng Lu, Kaicheng Shao, Weidi Sun, and Meng Sun. 2022. RGChaser: A RL-guided Fuzz and Mutation Testing Framework for Deep Learning Systems. In *2022 9th International Conference on Dependable Systems and Their Applications (DSA)*. IEEE, 12–23.
[39] Yixing Luo, Xiao-Yi Zhang, et al. 2021. Targeting requirements violations of autonomous driving systems by dynamic evolutionary search. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 279–291.
[40] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).
[41] Annibale Panichella, Fitsum Meshesha Kifetew, and Paolo Tonella. 2015. Reformulating branch coverage as a many-objective optimization problem. In *2015 IEEE 8th international conference on software testing, verification and validation (ICST)*. IEEE, 1–10.
[42] Prolific. 2024. General citation guidelines. Available at https://www.prolific.com. First released in 2014. Copyright 2024. Located in London, UK. Version: Current month(s) and year(s) of use..
[43] Luke Rowe, Roger Girgis, Anthony Gosselin, Liam Paull, Christopher Pal, and Felix Heide. 2025. Scenario dreamer: Vectorized latent diffusion for generating driving simulation environments. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 17207–17218.
[44] Shuo Sun, Zekai Gu, Tianchen Sun, Jiawei Sun, Chengran Yuan, Yuhang Han, Dongen Li, and Marcelo H Ang Jr. 2023. Drivescenegen: Generating diverse and realistic driving scenarios from scratch. *arXiv preprint arXiv:2309.14685* (2023).
[45] Inc. Tesla. 2024. Autopilot. https://www.tesla.com/en_AU/autopilot Accessed: 2024-11-13.
[46] Haoxiang Tian, Yan Jiang, et al. 2022. MOSAT: finding safety violations of autonomous driving systems using multi-objective genetic algorithm. In *ESEC/FSE 2022*. 94–106.
[47] Ziyuan Zhong, Gail Kaiser, and Baishakhi Ray. 2022. Neural network guided evolutionary fuzzing for finding traffic violations of autonomous vehicles. *IEEE Transactions on Software Engineering* (2022).

Received 2025-09-12; accepted 2026-03-24