ReGentS: Real-World Safety-Critical Driving Scenario Generation Made Stable

Yuan Yin[1], Pegah Khayatan[1*], Éloi Zablocki[1], Alexandre Boulch[1], and Matthieu Cord[1,2]

[1] Valeo.ai, Paris, France
{firstname.lastname}@valeo.com
[2] Sorbonne Université, CNRS, ISIR, F-75005 Paris, France

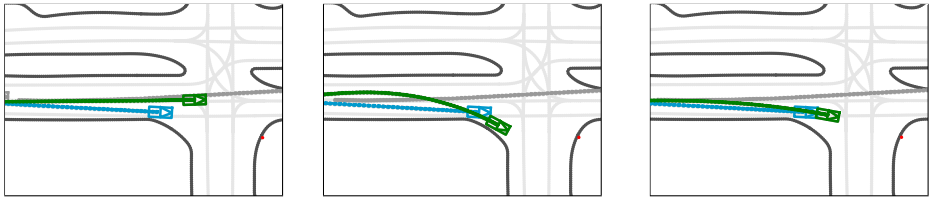
Abstract. Machine learning based autonomous driving systems often face challenges with safety-critical scenarios that are rare in real-world data, hindering their large-scale deployment. While increasing real-world training data coverage could address this issue, it is costly and dangerous. This work explores generating safety-critical driving scenarios by modifying complex real-world regular scenarios through trajectory optimization. We propose ReGentS, which stabilizes generated trajectories and introduces heuristics to avoid obvious collisions and optimization problems. Our approach addresses unrealistic diverging trajectories and unavoidable collision scenarios that are not useful for training robust planner. We also extend the scenario generation framework to handle real-world data with up to 32 agents. Additionally, by using a differentiable simulator, our approach simplifies gradient descent-based optimization involving a simulator, paving the way for future advancements. The code is available at https://github.com/valeoai/ReGentS.

Keywords: safety-critical scenario generation, corner case, trajectory optimization, robustness, bird's-eye-view

1 Introduction

With the rapid advancement of machine learning (ML), autonomous driving systems are becoming increasingly proficient, particularly through neural network-based methods [2]. However, their performance relies heavily on the extent of training data coverage, and they may struggle with unseen, safety-critical scenarios rarely encountered in real-world data. Improving out-of-distribution generalization remains a important topic within the ML community [16]. This challenge hinders the large-scale deployment of autonomous driving systems, which must meet strict safety standards. One solution is to enhance real-world data coverage by collecting more diverse data, such as through event data recorders (EDRs)

* Work done during an internship at Valeo.ai.



(a) Original **(b)** Without constraints [8] **(c)** Ours: ReGentS

Fig. 1: Importance of Constraints in Ego-Adversary Collision Scenarios. The figures depict the final state with past trajectories shown as dotted lines. The original collision-free scenario (a) is modified by an unconstrained method [8] (b) and our ReGentS (c) with proposed constraints. The ego vehicle is at the center, and the adversary is in front. In (b), the unconstrained adversary takes an unrealistic swinging turn (first to the left then to the right), leading to a partially diverging trajectory that does not reflect plausible real-world driving behavior. In contrast, our ReGentS (c) ensures the adversary decelerates smoothly while gradually approaching the ego, maintaining a realistic and natural trajectory.

[13]. However, collecting data specifically for safety-critical scenarios is dangerous, costly, and fraught with privacy concerns. Alternatively, generating these scenarios offers a solution. However, most existing approaches [8, 10, 12, 14, 15] have been benchmarked on synthetic data, which features scenarios of limited complexity and often involves few vehicles. Additionally, the generation process is often unconstrained, which may produce unrealistic solutions as shown in Fig. 1b or scenarios not useful for enhancing ML-based planner's robustness.

In this work, we introduce ReGentS, an approach for generating more stable safety-critical driving scenarios based on real-world data. We study some common choices in existing approaches, such as the cost that induces collisions. By focusing on KING [8] and refactoring it at a larger scale and in a more realistic setting with real-world data based simulator, Waymax [7], we identify following areas for improvement: (1) Unrealistic swinging trajectories may be produced, and (2) Many generated collisions involve the ego vehicle being rear-ended, which cannot in practice be avoided and thus does not provide useful corner cases for planner training. We address these issues by analyzing the optimization process of [8] and adding constraints to produce more stable trajectories. Additionally, we introduce a heuristic to avoid obvious collision scenarios. Our approach enhances stability of generated trajectories with more realistic driving behavior compared to the unconstrained method, as shown in Figs. 1b and 1c.

In a technical viewpoint, we implement and simplify the setup using a single differentiable simulator [7]. This allows for direct optimization through gradient descent, making the process simpler, unified, and easier to extend. The differentiable simulator also avoids the heavy interaction between a differentiable proxy simulator and non-differentiable one like CARLA [4], and eliminating inconsistencies that may arise between the two simulators.

In summary, our contributions are:

- We propose solutions to two main issues identified in existing approaches: solution stability and the choice of adversarial vehicles. Specifically, we adapt the optimization process to enhance stability with tailored analysis and introduce heuristics based on the position of the background vehicle.
- We implement a trajectory optimization-based scenario generation pipeline on a large-scale, differentiable simulator for real scenarios and data, facilitating its use in further studies.
- We demonstrate that ReGentS selects better adversaries and generates more stable trajectories in certain cases. Quantitatively, it produces more safety-critical scenarios, which is useful for fine-tuning ML-based planners.

The paper is organized as follows: Sec. 2 presents previous and related literature, Sec. 3 describes the addressed problem, Sec. 4 is a complete description of ReGentS and Sec. 5 is dedicated to experimental evaluation of our approach.

2 Related Work

The literature has been concentrating on trajectory generation to create ego-adversary collisions, which we categorize into the following approaches.

Black-box Optimization Approaches. Early methods treat the problem as a black-box adversarial attack. For example, AdvSim [14] uses Bayesian optimization. It employs a trajectory sampling method that rejects non-physically realizable trajectories, and proposes a cost function to bias the sampling to create collisions.

Generative Model-based Approaches. Generative models learn distributions based on provided data. In recent advances of complete scenario generation, such as [11], the results may lack diversity and physical feasibility. More specifically, in safety-critical case generation, these approaches typically involve two steps: (1) training a generative model to sample from a regular distribution, and (2) biasing the generation to create collisions using hand-crafted losses. STRIVE [12] uses a conditional variational autoencoder (CVAE) to learn a latent space of regular scenarios, which can then be adjusted to generate collisions. CAT [17] introduces a resampling strategy to bias generative models towards collision scenarios. Recent efforts, such as [10, 15], have introduced diffusion models that generate trajectories by diffusing states/actions from Gaussian noise. These models compute the gradient of a collision-inducing cost w.r.t. the states/actions to guide the diffusion process using classifier guidance.

Model-Based Trajectory Optimization Approaches. Unlike previous approaches, model-based methods such as KING [8] optimize trajectories directly by leveraging a kinematic model and actions.

The costs used by all the approaches above are derived from the collision condition eq. (2). They are mainly benchmarked on simulators with limited realism, such as CARLA [4]. Instead of injecting bias into a distribution as in (a) and (b), we choose to modify existing scenarios directly, as in (c). Unlike the unconstrained optimization in [8], we impose additional constraints.

3 Problem Setting

In this section, to put important elements in order, we provide a formal definition of safety-critical scenario generation common in the domain. In the following, without loss of generality, we focus on the bird's-eye-view (BEV) scenario.

3.1 Scenario Definition

We define a driving scenario $s \in \mathcal{S}$ as being composed of time-invariant contexts and time-variant series.

Time-invariant contexts include essential information and parameters for constructing the scenario background. This includes the road graph g, which specifies the positions of road features such as boundaries and lane center lines. Additionally, vehicle (agent) metadata $\{v^{(i)}\}_{i\in[\![0,n]\!]}$ is provided, where i indicates the i-th agent in the scenario. The metadata $v^{(i)}$ includes the width and length of agent's bounding box. The agent designated as $i = 0$ is the ego agent, while agents $i \in [\![1, n]\!]$ represent the n background (potentially adversarial) agents.

Time-variant series correspond to two categories: traffic signals and trajectories. Each series is uniformly discretized over T time steps $k \in [\![0, T-1]\!]$ with interval δt, meaning that k represent $k\delta t$. Traffic signals are a series $w = (w_k)_{k\in[\![0,T-1]\!]}$, including, e.g., traffic light statuses. The trajectories of the ego and background agents are denoted by $\{q_\pi^{(0)}\} \cup \{q^{(i)}\}_{i\in[\![1,n]\!]}$, where each $q^{(i)}$ is represented as a series of states, i.e., $q^{(i)} = (q_k^{(i)})_{k\in[\![0,T-1]\!]}$. The ego trajectory q_π^0 is unrolled with a planner π. Typically, each state q_k includes the position x_k, the speed u_k, and the orientation angle ψ_k, i.e., $q_k = (x_k, u_k, \psi_k) \in \mathcal{Q} = \mathbb{R}^2 \times \mathbb{R} \times [-\pi, \pi]$. To summarize, a scenario is defined as $s = (g, \{v^{(i)}\}, w, \{q_\pi^{(0)}\} \cup \{q^{(i)}\}_{i>0})$. By abuse of notation, we denote the scene at a time step k by $s_k = (g, \{v^{(i)}\}, w_k, \{q_{k,\pi}^{(0)}\} \cup \{q_k^{(i)}\}_{i>0})$.

3.2 Collision-Style Safety-Critical Condition

A widely accepted criterion of a "safety-critical" scenario in the field (see Sec. 2 for examples) is that there should be a collision between the ego agent $i = 0$ and any background agent $i > 0$. For a scenario s, it is commonly defined based on the distance between the bounding boxes of agents i and j at time step k:

$$d_{\mathrm{BB}}^s(i, j, k) = d\Big(\mathrm{BB}(q_{k,\pi}^{(i)}, v^{(0)}), \mathrm{BB}(q_k^{(j)}, v^{(i)})\Big) \tag{1}$$

where BB computes the agent's bounding box from the state q_k (position and orientation) and the agent's metadata $v^{(i)}$, and the distance d is the Euclidean distance in $\mathbb{R}^2$ between closest points on the bounding polygons. Therefore, a scenario is considered safety-critical if

$$\exists k \in [\![0, T-1]\!], i > 0, d_{\mathrm{BB}}^s(0, i, k) \leq 0. \tag{2}$$

indicating an overlap between the ego agent $i = 0$ and any adversary $i > 0$ at some time step k.

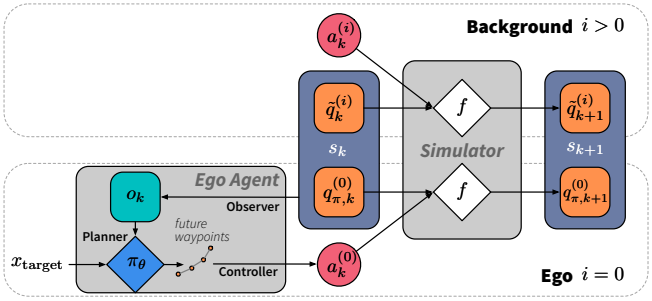


Fig. 2: Forward Model Assumption of ReGentS Through a Differentiable Simulator, accompanied by the agent's structure for state transition from s_k to s_{k+1}. The ego agent ($i = 0$) generates an observation o_k by rasterizing the scenario onto a uniform grid. The planner uses this observation and a target point x_{target} to predict the ego agent's action $a_k^{(0)}$ with a controller, which is fed into the kinematic model f. For background agents ($i > 0$), the model f estimates their actions $a^{(i)}$ and reconstructs their logged trajectory $\tilde{q}^{(i)}$. Arrows represent the computational graph which enables the backward autodiff.

3.3 Safety-Critical Scenario Generation with Regular Data

To generate scenario with collisions involving the ego agent, we use a dataset of scenarios $\mathcal{D} = \{s\}_{s\sim p_{\mathcal{S}}(s)}$, collected from the unknown real-world scenario distribution $p_{\mathcal{S}}$. These data are often retrieved using well-equipped vehicles driven by trained drivers. Publicly available datasets typically contain common scenarios under regular driving conditions, often lacking safety-critical situations.

However, generating the entire scenario s is difficult for safety-critical situations due to the lack of safety-critical reference data. In practice, scenario generation is further narrowed down to the conditional generation of trajectories with ego-adversary collisions. This involves generating adversary trajectories that are both plausible and satisfy the collision condition in Eq. (2), given the road graph g, agent metadata v, and potentially the traffic light statuses w.

4 Methodology

In this section, we present ReGentS as follows: We begin by introducing the model assumptions in Sec. 4.1. Next, we provide an overview of [8], the prerequisite of our work in Sec. 4.2. Finally, we describe our ReGentS, which results from our analysis of the optimization process, detailed in Sec. 4.3.

4.1 Model Assumption

The trajectory of each background agent ($i > 0$) is supposed to be controlled by a series of actions through a discretized kinematic dynamics model: $q_{k+1}^{(i)} = q_k^{(i)} + f(q_k^{(i)}, a_k^{(i)})\delta t$ where f is the vehicle's dynamics model (e.g., a Bicycle

Model considered in [8]), $a^{(i)} = (a_k^{(i)})_{k\in[\![0,T-2]\!]}$ is a series of $T-1$ actions. We denote this action-conditioned trajectory as $\tilde{q}^{(i)} = (q_0^{(i)}) \parallel (\tilde{q}_k^{(i)})_{k\in[\![1,T-1]\!]}$ where $\tilde{q}_k^{(i)} = q_0^{(i)} + \sum_{s=0}^{k-1} f(q_s^{(i)}, a_s^{(i)})\delta t$. The action-conditioned trajectory $\tilde{q}^{(i)}$ should recover the original one $q^{(i)}$, i.e., $\tilde{q}^{(i)} \approx q^{(i)}$. A scenario s is therefore extended as $s = (g, w, \{v^{(i)}\}, \{q_\pi^{(0)}\} \cup \{\tilde{q}^{(i)}\}_{i>0}, \{a^{(i)}\}_{i>0})$. Note that the action series $a^{(i)}$ are typically not available in most datasets and require to be estimated given the kinematic model f and original trajectories. The ego agent is assumed as a mapping $\pi\colon (s_k, x_{\text{target}}) \mapsto a_k$, which can be further decomposed into an observer-planner-controller structure (see details in Sec. 5.1). We depict this forward scheme in Fig. 2.

4.2 Prerequisite: KING

Our method extends KING [8], which is based on the setting described in Sec. 3 and generates new scenarios by modifying the trajectories of background agents in each $s \in \mathcal{D}$ using a kinematic model and estimated agent actions. The core of this approach is detailed below.

Scenario Generation as Optimization Problem. The scenario generation problem, as defined in KING, involves solving an optimization problem using a cost function $\mathcal{C}(s)$ for a given scenario s that encourages background agents to provoke a collision with the ego agent in an initially collision-free scenario. The optimization problem is formulated as:

$$a^* = \arg\min_a \mathcal{C}(s) \quad \text{s.t.} \quad \forall i \in [\![0,n]\!], k \in [\![0,T-2]\!], q_{k+1}^{(i)} = q_k^{(i)} + f(q_k^{(i)}, a_k^{(i)})\delta t \quad (3)$$

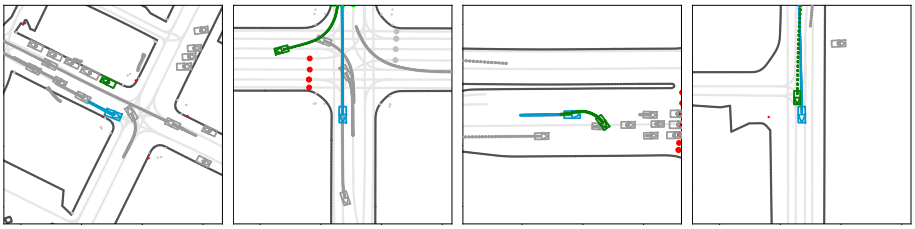
where a represents the action series of all background agents $\{a^{(i)}\}_{i>0}$. Solving this problem finds the actions that minimize the cost, subject to the physical dynamics model constraints.

Cost Definition. [8] decomposes the cost $\mathcal{C}$ into two parts: (a) *Collision Induction*: A cost $\mathcal{C}_{col}^{ego}$ is introduced to encourage collision between the ego agent and one of the background agents; (b) *Background Agent Regularization*: Two costs $\mathcal{C}_{col}^{adv}, \mathcal{C}_{dev}^{adv}$ are introduced to avoid collisions among background agents and prevent the background agents from deviating from the drivable area.

The cost (a) considers the average distance over all time steps $k \in [\![0, T-1]\!]$ between the ego agent ($i = 0$) and each background agent ($i > 0$) to indicate how likely a collision can be provoked:

$$\mathcal{C}_{col}^{ego}(s) = \min_{i\in[\![1,n]\!]} \frac{1}{T} \sum_{k=0}^{T-1} d_{\text{BB}}^s(0, i, k). \quad (4)$$

This function selects the closest background agent to minimize its distance to the ego. Regularization (b) is intended to avoid unwanted solutions. This first



(a) Minimum Trap: The optimization fixates to the nearest static adversary.

(b) Time-Averaged Distance Bias: Distance minimized at every time step without emphasis.

(c) Diverging Front Adversary: The distance minimization creates a trajectory that diverges from the ego's path.

(d) Converging Rear-End Adversary: The adversary is on a collision course to rear-end the ego agent.

Fig. 3: Illustration of Representative Issues with Unconstrained KING Method [8]. The final state of each scenario is shown along with the trajectory history in dotted line. The ego agent (blue) is at the center, and the adversary agent (green) represents the last optimized adversary.

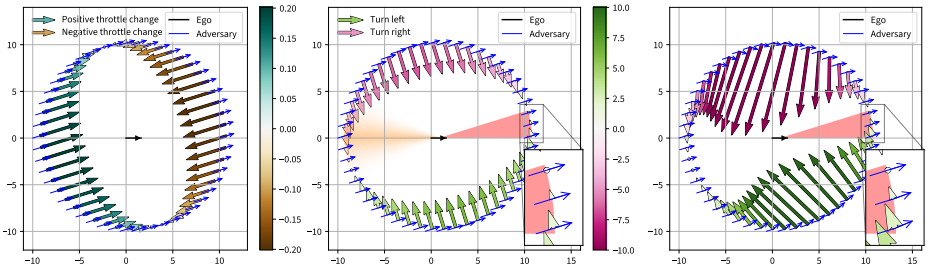
term discourages collisions between background agents using distance-based repulsion: $\mathcal{C}_{col}^{adv}(s) = -\min_{i,j\in[\![1,n]\!],k\in[\![0,T-1]\!]} \min(\tau, d_{\mathrm{BB}}^{s}(i,j,k))$, which measures the minimum distance between any pair of background agents over time, lower-bounded by threshold τ. The second term is intended to help background agents stay within the drivable area: $\mathcal{C}_{dev}^{adv}(s) = \frac{1}{T}\sum_{k=0}^{T-1}\sum_{i=1}^{n}\sum_{c\in\{c_l\}_{l=1}^{4}}(m_{oob} * \mathcal{K})\big(\mathrm{BB}\left(q_k^{(i)}, v_k^{(i)}\right)[c]\big)$. Here, $\mathcal{K}$ is a 2D Gaussian kernel, and $\{c_l\}_{l=1}^{4}$ represents the four corners of the agent's bounding box. The kernel is displaced by the coordinates of each corner and convolved with the binary drivable area map m_{oob}. A higher cost indicates a greater distance of background agents' bounding boxes from the drivable zone.

Solving Optimization Problem Using Gradient Descent. The optimization problem Eq. (3) is solved with gradient descent methods. The gradient of the cost function $\nabla_a \mathcal{C}(s)$ is computed w.r.t. all action series $\{a^{(i)}\}_{i>0}$ across all time steps, which is feasible if the kinematic model is differentiable. The optimization iteration is stopped once the condition Eq. (2) is satisfied.

4.3 ReGentS

Analysis. We focus on studying the main collision induction cost Eq. (4) and its optimization. We first provide an analysis on the optimization problem, the optimization algorithm and the kinematic gradient of the main cost function.

Optimization Bias. The current approaches to safety-critical scenario generation are based on a distance-based loss derived from the acceptance condition of Eq. (2). However, solving the optimization problem with the cost function Eq. (4) introduces some undesired biases:



(a) Negative gradient of distance cost w.r.t. throttle

(b) Negative gradient of distance cost w.r.t. steer

(c) Negative gradient of distance cost w.r.t. steer after one action update with the gradient shown in (a) and (b).

Fig. 4: Negative Gradient of the Distance Between the Ego Agent (→) and an Adversary (→) w.r.t. the Adversary's Actions. In (b), two interesting zones are highlighted. **Diverging Front Adversary** zone (red): Adversaries could pursue a diverging trajectory w.r.t. the ego agent. The negative gradient points to the diverging direction. The angle this zone is determined by the yaw of the adversary. **Converging Rear-end Adversary** zone (orange): Adversaries are likely to rear-end the ego agent. After one gradient descent update from (b) to (c), the negative gradient in the zoomed area intensifies, favoring the divergence of the adversary.

- *Minimum Trap*: Although Eq. (4) does not explicitly designate an adversary, the hard minimum operator can cause the optimization to fixate on the adversary chosen in the first iteration. As shown in Fig. 3a, the collision cost fixates on a stopped agent, while the agent turning at the intersection could have created a more interesting and avoidable collision.
- *Time-Averaged Distance Bias*: As shown in Fig. 3b, the green adversary intended to turn right is pulled back towards the ego by the minimization of the time-averaged distance. A simpler solution would be to optimize only the trajectory before the adversary arriving at the intersection. This showcases that minimizing distance over all time steps may lead to ineffective solutions.
- *Limited Diversity in Adversary Trajectory*: Optimizing the action series $(a_k)_k$ of the chosen adversary relies solely on the optimization algorithm and its hyperparameters (e.g., learning rate or Adam's decay rates), limiting the diversity of results.

Kinematic Gradient Bias. Defining a distance-based loss to create collisions is convenient, but it can also lead to unrealistic behaviors with methods rely on kinematic gradient information w.r.t. agent actions, due to constraints on the degrees of freedom (DoF) of actions. We illustrate some issues with KING [8], which relies heavily on kinematic gradients. Note that these issues may also affect recent classifier-conditioned diffusion models that generate actions to produce trajectories with a kinematic model [10, 15], using the same gradient to guide the generative model to induce collisions.

To illustrate the issue in the kinematic gradient, we present a simple scenario involving only the ego agent and a single adversary agent to be optimized. The adversary is shown at different locations but with the same distance to the ego, as shown in Fig. 4. Both agents are moving at a constant speed to the right. The key difference is their yaw: the ego's yaw is aligned with the x-axis, while the adversary's yaw is slightly offset. This scenario is particularly interesting when using real-world data because even if background agents are heading in the same direction as the ego, their yaws cannot be perfectly aligned due to noise in the data. The yaw offset shown in Fig. 4 is chosen for better illustration and visualization.

In Fig. 4, we show the gradient descent directions (i.e., negative gradient) for the acceleration (Fig. 4a) and steering (Fig. 4b) of the adversary calculated from the distance between the ego and the adversary. As shown in Fig. 4a, the negative gradient of acceleration aligns well with the intuition of a typical driver: the adversary diminish its acceleration if it is ahead of the ego; it accelerates more if it is behind. However, the steering negative gradient, shown in Fig. 4b, is more complex. We focus on two specific zones for the adversary (red and orange), where undesired situations may occur. The angle of these zones corresponds to the difference in orientations between the ego and the adversary.

- *Diverging Front Adversary*: As illustrated in Fig. 3c, the agent could have created a collision with the ego by just decelerating, while it did unnecessary turns which induced its collision with another background agent. The reason why this happens is when an adversary enters the red zone in Fig. 4b its negative gradient suggests it should turn away from the ego, causing the adversary's trajectory to diverge from the ego's. Subsequently, updating the actions with this negative gradient (see Fig. 4c) amplifies the yaw difference for agents already in the red zone, creating larger negative gradients which could push them further onto a more diverging trajectory (see the zoomed area in Figs. 4b and 4c). As a result, agents in this zone are prone to showing unrealistic divergent behaviors, such as swinging turns between left and right.
- *Converging Rear-End Adversary*: In real-world driving environments, rear-end collisions are less relevant for improving the robustness of the planner. For example, if the ego vehicle is in dense traffic, it is almost impossible to avoid such a collision. However, optimizing adversaries in some areas is doomed to create a rear-end collision. As shown in Fig. 3d, when an adversary is behind the ego, i.e., in the orange zone in Fig. 4b, it will be attracted to align with the yaw of the ego. It is very unlikely that the adversary in this area can overtake the ego because the tendency to diverge from the ego's past trajectory will be counteracted. Therefore, the optimized adversary is very likely to rear-end the ego, leading to uninteresting cases.

ReGentS' Remedy. Inspired by the previous analysis, we propose ReGentS, which consists of several constraint rules designed to address some of the identified issues. Our goal is to make these criteria as flexible as possible.

Stop Front Divergence. As discussed in Sec. 4.3, front divergence may be caused by an aggrieving steer negative gradient favoring diverging trajectories. To mitigate this, we establish the following rule: we cancel the update for the steering action if an adversary stays (i) *ahead of the ego agent* and (ii) its *yaw is offset to the same side as its position relative to the ego* during a certain percentage of time steps, defined by a threshold τ_{front}.

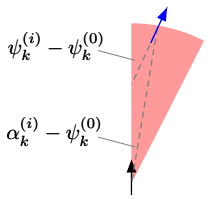


Fig. 5: Illustration of how to determine if an adversary is in the red zone, with the red zone calculation shown. The ego agent is indicated by (→) and the adversary by (→).

The rule is illustrated in Fig. 5, featuring the same red zone shown in Fig. 4. At each time step k, the angle of the red zone is calculated by $\psi_k^{(i)} - \psi_k^{(0)}$. The deviation of the adversary's angular position relative to the ego's orientation, denoted by $\alpha_k^{(i)} - \psi_k^{(0)}$, is compared to the red zone angle to determine if the adversary is within the red zone at time k. This check is performed only for background agents with a yaw difference $\psi_k^{(i)} - \psi_k^{(0)} \in (-\frac{\pi}{8}, \frac{\pi}{8})$ and whose angular position is $\alpha_k^{(i)} - \psi_k^{(0)} \in (-\frac{\pi}{8}, \frac{\pi}{8})$ in front of the ego agent.

Exclude Rear-End and Static Adversaries. As explained in Sec. 4.3, we may want to avoid rear-end collisions, which are unavoidable for the ego vehicle in some cases. In ReGentS, we consider collisions in the orange zone as non-actionable and choose to exclude adversaries situated in this zone if they remain there at most time steps in the original scenario. The central angle of the disk sector zone is adjusted empirically as a hyperparameter. In our experiments, we set this zone as large as 45 degrees, with its axis of symmetry aligned with the x-axis.

To avoid the problem where the optimization stick to a static adversary, we simply exclude all the static adversaries in the scenario from candidate to the optimization.

5 Experimental Results

5.1 Implementation for Real-World Scenarios

Data and Simulator. In our work, we uses the Waymo Open Motion Dataset (WOMD) [6] and Waymax [7], a differentiable simulator based on the scenario data in WOMD. The dataset comprises the scenarios recorded in cities, and we limit the maximum number of agents in a scenario to 32. The main advantage of Waymax is its implementation with JAX [1], which is inherently differentiable, in contrast to CARLA, eliminating the need for another differentiable proxy simulator. In comparison, [8] uses a synthetic dataset featuring up to four background agents alongside the non-differentiable CARLA simulator. This setup significantly limited the flexibility of gradient calculations necessary to optimize trajectories through actions, thus necessitating a differentiable proxy simulator as a workaround.

Neural Network-Based Ego Agent. We employ a neural network-based ego agent tailored for WOMD. The agent $\pi\colon (s_k, x_{\text{target}}) \mapsto a_k$ determines actions based on the current scene s_k at time step k and a target point x_{target}. We adapt the AIM-BEV agent from KING, consisting of the following components: (1) **Observer**: An observer $\text{Obs}\colon s_k \mapsto o_k$ is implemented for WOMD to provide a rasterized ego-centered BEV observation o_k, oriented to ego's yaw direction. (2) **Planner**: Using MobileNet-V3 [9], the observation o_k is transformed into hidden features h_k via $\text{Enc}\colon o_k \mapsto h_k$. A gated recurrent unit (GRU) [3] cell is used to predict L future waypoints $(x_{k+l})_{l\in[\![1,L]\!]}$. At each time step $k+l$, the GRU cell updates its hidden state to $h_{k+l+1} = \text{GRU}(h_{k+l}, [x_{k+l}, x_{\text{target}}, x_{\text{target}} - x_{k+l}])$, using position x_{k+l}, target point x_{target}, and their difference as inputs. The hidden state is then decoded by $\text{Dec}(\cdot)$ to predict the displacement towards the next waypoint $x_{k+l+1} = x_{k+l} + \text{Dec}(h_{k+l+1})$. (3) **Controller**: We follow [8] by using PID controller to estimate ego actions from the predicted waypoints $(x_{k+l})_{l\in[\![1,L]\!]} \mapsto a_k$. The planner (2) is trained to learn to map from (o_k, x_{target}) to future waypoints $(x_{k+l})_{l\in[\![1,L]\!]}$. Training is done via imitation learning using supervised input-output examples of this mapping.

Action Estimation for Background Agents. As mentioned in Sec. 4.2, in real-world scenarios, only a series of waypoint coordinates $(x_k)_k$ is known and the actions must be estimated. We use an invertible bicycle model to estimate actions using local derivatives, accurately recovering physically plausible trajectories. Note that this approach may produce noisy actions due to data noise; we leave addressing this issue to future work.

Simulation and Optimization. We simulate the scenarios using the AIM-BEV planner for the ego vehicle and estimated actions for the background agents. By leveraging Waymax, we can efficiently compute the exact gradient of the cost w.r.t. the actions throughout the entire trajectory. This is in contrast to the approach used in [8], which uses an approximate gradient to reduce calculation overhead by stopping gradients at specific moments. For both ReGentS and [8], we use the Adam optimizer with a learning rate of 10^{-3} and a maximum of 500 optimization iterations. The optimization process is unsuccessful if it fails to generate a collision.

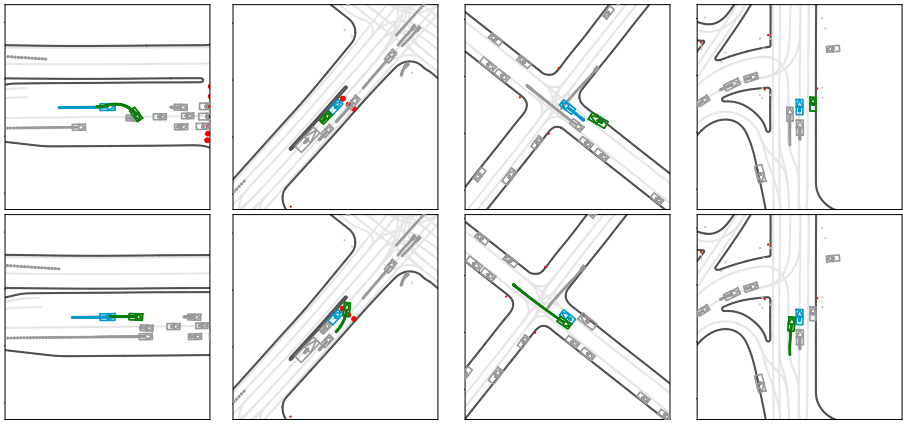
5.2 Effectiveness of ReGentS

Evaluation Protocol. We conducted our experiments using 200 scenarios extracted from the validation set of WOMD [6]. First, we validated the ego agent using the original scenarios. Then, we generated safety — critical scenarios with ReGentS and KING and compared their effectiveness both qualitatively — illustrated with examples — and quantitatively — based on the number of scenarios successfully generated and their impact on the ego agent's performance.

Ego Agent Validation. Since our objective is to enhance an ego agent, we start with one that can drive correctly in regular scenarios. The following scores of the ego planner are reported for these scenarios:

- **Road Completion** measures the average percentage of the route that the planner completes by the horizon of each scenario or before collision.
- **Infraction Score** is an accumulated penalty when the ego agent collides with another agent, a cyclist, a pedestrian, or ran out of the road boundary. We implemented the score for WOMD with the same penalty rate as [8].
- **Driving score** is the product of the two previous scores.
- **Collision rate** is the rate of collisions of the ego agent.

Our trained agent can reach given target points with an average route completion rate of 88.05%, indicating acceptable performance. However, despite the high route completion rate, the collision rate is 25.50%.



(a) ReGentS removes diverging behavior **(b)** ReGentS avoids rear-end **(c)** ReGentS ignores static vehicles **(d)** ReGentS ignores static vehicles and avoids rear-end

Fig. 6: Qualitative Comparison of Generated Scenarios. The first row shows the solutions provided by KING [8], while the second row displays ReGentS' results. The final state of each scenario is shown along with the trajectory history in dotted line. The ego agent (blue) is at the center, and the adversary agent (green) represents the last optimized adversary. Our method provides non-divergent trajectories and choose better adversary to create more interesting cases.

Qualitative Results. In Fig. 6, we present a qualitative comparison of ReGentS with KING. In Fig. 6a, [8] makes the optimized adversary run into another background vehicle, despite regularization, while ReGentS can steadily decelerate the adversary without creating diverging trajectories. In Fig. 6b, ReGentS selects a background agent that provokes a merging collision, in contrast to [8], which results in a less interesting rear-end collision. In Fig. 6c, ReGentS removes the static agent and creates a head-on collision with the oncoming agent, whereas KING focuses on a stopped vehicle and attempts to start it from the stopping state. In Fig. 6d, ReGentS ignores the stopping agent on the right and the other

Table 1: Comparison of Collision Generation Effectiveness for ReGentS and [8]. The table presents the ego agent's performance scores across 200 original WOMD validation scenarios, as well as those modified by ReGentS and [8], alongside their generation success rates. "↑ for generation" indicates that a higher value is better for collision generation, vice versa. The metrics are defined in the beginning of Sec. 5.2.

WOMD Val Scenarios		Original	KING [8]	ReGentS
Generation Success Rate (↑ for generation)		n/a	48.99	**60.40**
Ego Agent Performance	Route Completion (↑ for ego, ↓ for generation)	88.05	73.68	71.06
	Infraction Score (↑ for ego, ↓ for generation)	0.82	0.72	0.69
	Driving Score (↑ for ego, ↓ for generation)	74.20	56.19	50.78
	Collision Rate (↓ for ego, ↑ for generation)	25.50	59.00	69.50

agent behind with risk of rear-end, and modifies marginally the agent on the left to create a lane merging collision from the right side. These qualitative results indicate better stability in adversary trajectory generation and improved adversary selection in ReGentS compared to [8].

Quantitative Results. To demonstrate the effectiveness of collision generation, we report **Generation Success Rate**, the percentage of collisions that corresponding method have successfully created among the previously collision-free scenarios. In Tab. 1, we present the results comparing scenarios modified by ReGentS with those generated by the original KING.

We observe a significant drop in the collision generation success rate for KING, decreasing from approximately 80% with synthetic scenarios in the original work to 49% with WOMD. This decline highlights the challenges of applying this method to real-world settings. In contrast, ReGentS improves the success rate to 60% with WOMD, representing a 10 percentage point increase.

Additionally, both ReGentS and KING generate collisions that negatively impact the ego vehicle's driving performance. ReGentS tends to penalize the ego agent more, likely due to the higher number of collisions created.

5.3 Notes for Implementation

Implementing our approach with Waymax and JAX required reworking most components from scratch to ensure efficient optimization.

We leveraged just-in-time (JIT) compilation and function vectorization to parallelize and accelerate most parts of the calculation for optimal performance on GPUs. New functions were implemented for rasterizing observations directly on GPUs, accelerating the ego agent's planning and scenario simulation.

Due to the lack of some information in the WOMD dataset, cost terms were adapted based on available information. For example, no drivable area map was directly provided in WOMD, so we rebuilt it using road boundary information. For the same reason, some evaluation metrics were adapted, such as the out-of-road penalty score, which requires access to the drivable area map.

Additionally, we smoothly integrated a pretrained neural network with PyTorch into our JAX pipeline using `torch2jax` [5], avoiding potential performance issues caused by network architecture re-implementation and weight transfer.

6 Conclusion

In this work, we propose ReGentS, which enhances the stability of kinematic model-based trajectory optimization methods. We address issues discovered after scaling the approach from [8] for use in real-world settings. Our method identifies and mitigates issues that cause instabilities and generate unactionable scenarios. It generates more meaningful scenarios in complex settings and provides a robust and flexible technical foundation for developing efficient solutions for safety-critical scenario generation. In future work, it would be valuable to investigate how well the generated corner-cases, which are currently optimized to a specific ego agent's planning algorithm, generalize to different driving agents. Additionally, exploring whether it is possible to design corner-cases that universally cause planners to fail would be an interesting avenue to pursue.

Acknowledgements

This work was supported by the ANR grant MultiTrans (ANR-21-CE23-0032). We thank the anonymous reviewers for their remarks. This work was made using the Waymax Licensed Materials, provided by Waymo LLC under the Waymax License Agreement for Non-Commercial Use, available at https://github.com/waymo-research/waymax/blob/main/LICENSE, and your access and use of the Waymax Licensed Materials are governed by the terms and conditions contained therein.

References

1. Bradbury, J., Frostig, R., Hawkins, P., Johnson, M.J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., Zhang, Q.: JAX: composable transformations of Python+NumPy programs (2018), http://github.com/google/jax
2. Chen, L., Wu, P., Chitta, K., Jaeger, B., Geiger, A., Li, H.: End-to-end autonomous driving: Challenges and frontiers (2024)
3. Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL. pp. 1724–1734. ACL (2014)
4. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: Proceedings of the 1st Annual Conference on Robot Learning. pp. 1–16 (2017)

5. Dyro, R.: torch2jax. https://https://github.com/rdyro/torch2jax (2023)
6. Ettinger, S., Cheng, S., Caine, B., Liu, C., Zhao, H., Pradhan, S., Chai, Y., Sapp, B., Qi, C.R., Zhou, Y., Yang, Z., Chouard, A., Sun, P., Ngiam, J., Vasudevan, V., McCauley, A., Shlens, J., Anguelov, D.: Large scale interactive motion forecasting for autonomous driving: The Waymo Open Motion Dataset. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 9710–9719 (October 2021)
7. Gulino, C., Fu, J., Luo, W., Tucker, G., Bronstein, E., Lu, Y., Harb, J., Pan, X., Wang, Y., Chen, X., Co-Reyes, J.D., Agarwal, R., Roelofs, R., Lu, Y., Montali, N., Mougin, P., Yang, Z., White, B., Faust, A., McAllister, R., Anguelov, D., Sapp, B.: Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. In: Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (2023)
8. Hanselmann, N., Renz, K., Chitta, K., Bhattacharyya, A., Geiger, A.: King: Generating safety-critical driving scenarios for robust imitation via kinematics gradients. In: European Conference on Computer Vision. pp. 335–352. Springer (2022)
9. Howard, A., Pang, R., Adam, H., Le, Q.V., Sandler, M., Chen, B., Wang, W., Chen, L., Tan, M., Chu, G., Vasudevan, V., Zhu, Y.: Searching for mobilenetv3. In: 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019. pp. 1314–1324. IEEE (2019)
10. Huang, Z., Zhang, Z., Vaidya, A., Chen, Y., Lv, C., Fisac, J.F.: Versatile scene-consistent traffic scenario generation as optimization with diffusion (2024)
11. Pronovost, E., Ganesina, M.R., Hendy, N., Wang, Z., Morales, A., Wang, K., Roy, N.: Scenario diffusion: Controllable driving scenario generation with diffusion. In: Advances in Neural Information Processing Systems. vol. 36, pp. 68873–68894. Curran Associates, Inc. (2023)
12. Rempe, D., Philion, J., Guibas, L.J., Fidler, S., Litany, O.: Generating useful accident-prone driving scenarios via a learned traffic prior. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022. pp. 17284–17294. IEEE (2022)
13. Tesla Inc.: Disclamers - Event Data Recorder (EDR) (2024), https://www.tesla.com/ownersmanual/model3/en_us/GUID-2E8E5E0B-DAA8-40B8-9804-45F5960538DF.html
14. Wang, J., Pun, A., Tu, J., Manivasagam, S., Sadat, A., Casas, S., Ren, M., Urtasun, R.: AdvSim: Generating safety-critical scenarios for self-driving vehicles. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021. pp. 9909–9918. Computer Vision Foundation / IEEE (2021)
15. Xu, C., Zhao, D., Sangiovanni-Vincentelli, A., Li, B.: Diffscene: Diffusion-based safety-critical scenario generation for autonomous vehicles. In: The Second Workshop on New Frontiers in Adversarial Machine Learning (2023)
16. Yang, J., Zhou, K., Li, Y., Liu, Z.: Generalized out-of-distribution detection: A survey. CoRR **abs/2110.11334** (2021)
17. Zhang, L., Peng, Z., Li, Q., Zhou, B.: Cat: Closed-loop adversarial training for safe end-to-end driving. In: Proceedings of The 7th Conference on Robot Learning. Proceedings of Machine Learning Research, vol. 229, pp. 2357–2372. PMLR (06–09 Nov 2023)