ARGUS: Resilience-Oriented Safety Assurance Framework for End-to-End ADSs

Dingji Wang*, You Lu*, Bihuan Chen*†, Shuo Hao*, Haowen Jiang*, Yifan Tian*, Xin Peng*
*College of Computer Science and Artificial Intelligence, Fudan University, Shanghai, China
†Corresponding Author

***Abstract*—End-to-end autonomous driving systems (ADSs), with their strong capabilities in environmental perception and generalizable driving decisions, are attracting growing attention from both academia and industry. However, once deployed on public roads, ADSs are inevitably exposed to diverse driving hazards that may compromise safety and degrade system performance. This raises a strong demand for resilience of ADSs, particularly the capability to continuously monitor driving hazards and adaptively respond to potential safety violations, which is crucial for maintaining robust driving behaviors in complex driving scenarios.**

To bridge this gap, we propose a resilience-oriented runtime framework, named ARGUS, to mitigate the driving hazards, thus preventing potential safety violations and improving the driving performance of an ADS. ARGUS continuously monitors the trajectories generated by the ADS for potential hazards and, whenever the EGO vehicle is deemed unsafe, seamlessly takes control via a hazard mitigator. We integrate ARGUS with three state-of-the-art end-to-end ADSs, *i.e.,* TCP, UniAD and VAD. Our evaluation has demonstrated that ARGUS effectively and efficiently enhances the resilience of ADSs, improving the driving score of ADSs by 150.30% on average, and preventing 64.38% of the violations, with little additional time overhead.

I. INTRODUCTION

Recently, the superior environmental perception and generalizable decision-making capabilities of end-to-end autonomous driving systems (ADSs), enabled by specialized model architectures and large-scale diverse training data [35, 41, 60], have attracted substantial investment from both academia and industry [35, 41, 75]. Unlike traditional modular pipelines [5], end-to-end ADSs do not rely on high-definition maps or extensive hand-crafted rules to handle diverse driving scenarios. These ADSs promise to enhance road safety, mitigate traffic congestion, and boost overall transportation efficiency, thereby catalyzing transformative change across the automotive industry [53]. However, they lack explicit logic explainability and safety boundaries compared to modular ADSs. Scenario-based testing approaches [18, 49, 50, 83] demonstrate that the ADSs may still suffer from safety violations even in scenarios that are highly similar to those in the training dataset, compromising safety and degrading system performance, as highlighted by numerous documented incidents [3, 23]. Therefore, it is important to enhance the resilience of ADSs, achieving the hazard mitigation while maintaining their advantages in interactive scenarios.

Software resilience focuses on the capability of systems to adapt to and recover from unexpected events while maintaining effective operation under hazardous conditions [33, 56]. Improving the resilience requires not only monitoring hazards but also building systems that can proactively mitigate hazards, recover quickly, and continue to operate effectively in complex conditions [59]. Several studies have explored approaches to enhance resilience across diverse domains, including aviation, robotics, and connected autonomous vehicle platoons [10, 22, 71].

As to end-to-end ADSs, only a few works [7, 19, 27] focus on unexpected condition prediction [63, 64] or rule-based misbehavior identification of ADSs [11, 12, 57, 80], but fail to provide a runtime solution for hazard mitigation. To prevent safety violations, several works [66, 67] make intrusive modifications to original ADSs and refine the trajectories generated by ADSs to ensure the specification compliance. Furthermore, various fallback strategies [36, 77, 79], *i.e.,* emergency braking [44, 82] and human-initiated takeovers [26, 81], have been proposed to ensure safety. These approaches primarily emphasize emergency interventions rather than enabling ADSs to automatically adapt and maintain effective operation during hazardous scenarios. However, according to the SAE levels [38] of automation, the fallback responsibility no longer lies with human drivers but instead must be handled by the ADS itself for Level 4+ [79]. To the best of our knowledge, there is no existing work that explicitly targets enhancing the resilience of end-to-end ADSs, particularly the capability to continuously monitor hazards, adaptively respond to potential safety violations, and recover quickly to sustain safe operation in hazardous scenarios.

To fill this gap, we present ARGUS, a resilience-oriented runtime framework that proactively prevents safety violations and enhances the driving performance of an end-to-end ADS. ARGUS comprises three components, *i.e.,* the *Takeover Gate*, the *Hazard Monitor*, and the *Hazard Mitigator*. Every trajectory generated by the ADS passes through the *Takeover Gate*, which is responsible for dynamic control switching between the ADS and the *Hazard Mitigator* that is built upon the intelligent driver model (IDM) [70]. The *Takeover Gate* leverages the takeover and recovery buffers maintained by the *Hazard Monitor* to determine whether a given trajectory is safe for execution, thereby realizing a hazard-aware takeover and recovery mechanism. If the EGO vehicle (*i.e.,* the vehicle controlled by the ADS) is deemed unsafe, control is taken over by the *Hazard Mitigator* for hazard mitigation, and is only returned to the ADS once safety is reestablished.

We have conducted large-scale experiments to evaluate the effectiveness and efficiency of ARGUS. First, we integrate ARGUS with three state-of-the-art end-to-end ADSs (*i.e.,* TCP [75], UniAD [35] and VAD [41]), and evaluate them

2643-1572/25/$31.00 ©2025 IEEE
DOI 10.1109/ASE63991.2025.00010

on two benchmarks (*i.e.,* Bench2Drive [39] and CARLA leaderboard 2.0 validation set [13]) with realistic perception-based BEVs and the ideal privileged environment information, respectively. Our experiments have demonstrated that ARGUS enhances the resilience of ADSs, improving the driving score of ADSs by 150.30% on average, with little additional time overhead. Besides, ARGUS is able to prevent 64.38% of the violations with better driving skills. Second, we determine the accuracy of these takeovers triggered by ARGUS, and the results has shown that ARGUS achieves a high F_3 score of 0.899 in producing takeover decisions. Then, we conduct ablation studies to assess the individual contribution of each step within the IDM-based hazard mitigator, followed by the parameter sensitivity analysis to evaluate the robustness of ARGUS in different parameter settings. Finally, we generalize ARGUS to Apollo [5], a modular ADS, where ARGUS achieves a 113.92% improvements over the original Apollo.

Overall, ARGUS introduces a modular, resilience-oriented safety assurance framework for AI-enabled complex systems with probabilistic uncertainty, embodying key software engineering principles (*i.e.,* runtime monitoring and adaptive recovery). The main contributions of our work are as follows.

- We propose a hazard-aware takeover and recovery mechanism and an IDM-based hazard mitigator to prevent safety violations and enhance the performance of end-to-end ADSs.
- We design and implement a framework ARGUS to enhance the resilience of end-to-end ADSs under driving hazards.
- We integrate ARGUS with three state-of-the-art end-to-end ADSs, and conduct experiments on two benchmarks to demonstrate the effectiveness and efficiency of ARGUS.

II. BACKGROUND

A. End-to-End Autonomous Driving Systems

End-to-end ADSs [15, 68] unify perception, prediction, and planning within a single model. These models process multi-modal sensor inputs (*e.g.,* multi-view camera images and radar), navigation points, and vehicle states (*e.g.,* speed, heading, and position) to plan future trajectories composed of short-term waypoints and the desired speed. A vehicle controller, typically a PID-based controller [8], subsequently converts every trajectory generated by the model into control commands (*i.e.,* brake, steer and throttle) to drive the vehicle.

Beyond performance improvement achieved by simply scaling training resources, recent work has proposed in-model safety mechanisms to enhance the robustness of end-to-end ADSs. For instance, Interfuser [60] formulates a linear program to optimize expected speed for collision avoidance. UniAD [35] employs Newton-based optimization over occupancy grids to generate collision-free trajectories. In addition, VAD [41] introduces instance-level planning constraints to improve planning reliability. However, recent studies [39, 62] have shown that end-to-end ADSs still lack resilience, with violations occurring even in scenarios encountered during training.

B. Intelligent Driver Model

The intelligent driver model (IDM) [70] is a widely used car-following model that computes positions and speeds based on a leading actor in front, typically serving as a component within rule-based driving models [1, 43]. For the EGO vehicle E and the selected leading actor $lead$, p_E denotes the position of E at time t, and v_E denotes its speed. Furthermore, s_E is defined as the net distance between E and $lead$, and Δv_E represents their relative speed. For a simplified version of the model, the dynamics of E are described by Eq. 1,

$$\begin{aligned} &\dot{p}_E = \frac{\mathrm{d}p_E}{\mathrm{d}t} = v_E \\ &\dot{v}_E = \frac{\mathrm{d}v_E}{\mathrm{d}t} = a\left(1 - \left(\frac{v_E}{v_0}\right)^{\sigma} - \left(\frac{s^*(v_E, \Delta v_E)}{s_E}\right)^2\right) \\ &\text{with } s^*\left(v_E, \Delta v_E\right) = s_0 + v_E T + \frac{v_E \Delta v_E}{2\sqrt{ab}} \end{aligned} \tag{1}$$

where v_0 denotes the desired speed, s_0 is the minimum desired net distance, T is the expected minimum time required for E to reach the position of $lead$, a is the maximum acceleration, b is the comfortable braking deceleration, and σ is the acceleration exponent that is usually set to 4.

Although the IDM has been extensively validated and is known for its effectiveness in modeling realistic traffic flow and longitudinal driving behavior, it remains limited in its applicability to hazard mitigation. In particular, its reliance on a single fixed leading actor and the absence of dynamic route adaptation limit its capability to navigate diverse driving hazards.

III. METHODOLOGY

We propose ARGUS to enhance the resilience of an end-to-end ADS under hazardous driving scenarios. Our framework targets three primary hazards, *i.e.,* the *collision hazard* where the EGO vehicle collides with traffic participants, the *stop signal hazard* where the EGO vehicle violates the stop signals (*e.g.,* stop signs and red lights), and the *stalling hazard* where the EGO vehicle is stalled in a dangerous situation. The overall idea of ARGUS is to asynchronously monitor the trajectories generated by the ADS for potential hazards, and to take over control using an IDM-based hazard mitigator when the EGO vehicle is deemed unsafe. Control is returned to the ADS only after the safety is reestablished.

A. Approach Overview

Fig. 1 shows the approach overview of ARGUS, which consists of three components, *i.e.,* the *Takeover Gate*, the *Hazard Monitor*, and the IDM-based *Hazard Mitigator*. Every trajectory (*i.e.,* the desired waypoints and speed) generated by the ADS passes through the *Takeover Gate* (see Sec. III-B), which checks three takeover buffers and one recovery buffer maintained by the *Hazard Monitor*, to determine whether it is safe for execution by a takeover and recovery mechanism. If the EGO vehicle is unsafe, the *Hazard Mitigator* is activated for mitigation.

The *Hazard Monitor* is responsible for asynchronous hazard detection based on trajectories generated by the ADS (see Sec. III-C). Specifically, it utilizes the Bird's Eye View (BEV) representations derived from sensor data (*i.e.,* multi-view camera images) to predict the motions of the EGO vehicle itself as

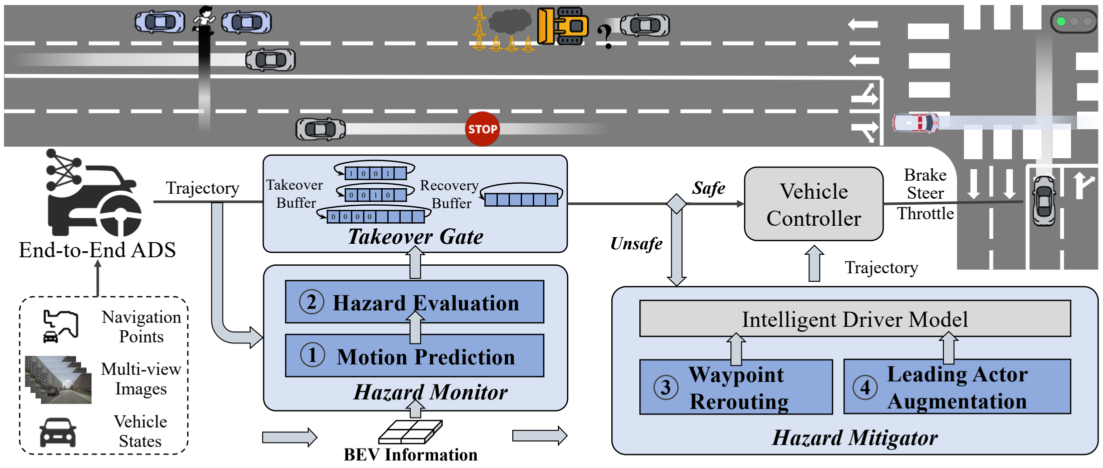


Fig. 1: Approach Overview of ARGUS

well as the surrounding traffic participants (*i.e.,* step ① in Fig. 1). Subsequently, the *Hazard Monitor* evaluates the potential hazards using the predicted motions and EGO vehicle states (*i.e.,* step ② in Fig. 1), updating the values of the three takeover buffers and one recovery buffer for the *Takeover Gate*.

The IDM-based *Hazard Mitigator* is responsible for performing active hazard mitigation according to the current hazardous situation (see Sec. III-D). Specifically, it generates and optimizes the waypoints to avoid obstacles (*i.e.,* step ③ in Fig. 1), subsequently passing the optimized waypoints to the IDM. Then, the *Hazard Mitigator* augments the leading actors along with the optimized waypoints (*i.e.,* step ④ in Fig. 1) for the IDM to compute the speed of the EGO vehicle and achieve dynamic route adaptation. Finally, the resulting trajectory, composed of the optimal waypoints and speed, is sent to the vehicle controller for execution, thereby circumventing the potential hazards.

B. Takeover Gate

To determine whether the control of the EGO vehicle should remain with the ADS or be temporarily taken over, we introduce the *Takeover Gate*. It checks three takeover buffers and one recovery buffer, deciding whether to dispatch the trajectory generated by the ADS to the vehicle controller.

Takeover Buffer. Implemented as a fixed-length circular queue, the takeover buffer stores the most recent hazard evaluation results for ADS trajectories. It provides a running window for identifying whether the ADS is currently exposed to hazards. Two distinct queues with length M are maintained for collision hazard and stop signal hazard, respectively. Besides, a queue with length N is maintained for stalling hazard.

Recovery Buffer. Likewise, realized as a circular queue of length R, the recovery buffer records the latest R hazard evaluations after a takeover. It tracks the ADS's progression from a hazardous state to a safe state, thereby determining when the control can be safely returned to the ADS.

Takeover and Recovery Mechanism. The takeover buffers and the recovery buffer are initially empty, and each of their entries is a binary value (*i.e.,* 0 or 1), denoting the absence or presence of a potential hazard. The values in these buffers are continuously updated and maintained by the *Hazard Monitor*, which will be introduced in Sec. III-C. Leveraging the entries in these buffers, the *Takeover Gate* determines whether a given trajectory should be executed by the vehicle controller.

Specifically, if the number of entries with value 1 in either the collision hazard buffer or stop signal hazard buffer exceeds a threshold l, or all entries in the stalling hazard buffer are 1, the current trajectory is not dispatched to the vehicle controller, and the downstream *Hazard Mitigator* is instead activated, which will be introduced in Sec. III-D. The collision and stop signal hazards are safety-critical, and thus a threshold-based conservative takeover strategy is adopted to account for potential false negatives in *Hazard Monitor*. The stalling hazard indicates that the EGO vehicle is persistently unable to proceed and therefore requires takeover. Once the control has been taken over, the recovery buffer is reset and begins recording the evaluation results for subsequent ADS trajectories. When all entries in the recovery buffer are 0, indicating that the ADS has remained in a consistently safe state and is capable of resuming the driving task, the control is returned to the ADS by allowing its newly generated trajectory to be dispatched to the vehicle controller.

To balance responsiveness and stability, we construct a small calibration set from [39], and empirically tune the parameters M, R and l to adapt to different ADS configurations.

C. Hazard Monitor

To detect the potential driving hazards (*i.e.,* the collision hazard, the stop signal hazard, and the stalling hazard), for each trajectory generated by the ADS, we asynchronously assess whether the ADS falls under potential hazards. This process involves two key steps, *i.e.,* the **motion prediction**, which prepares the predicted future motions of the EGO vehicle and surrounding traffic participants, and the **hazard evaluation**, which evaluates the potential hazards and maintains the takeover buffers and the recovery buffer.

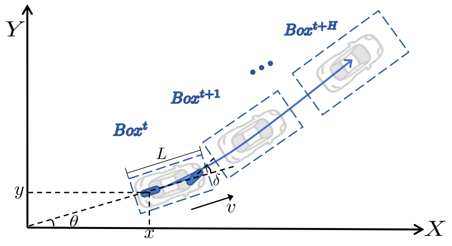


Fig. 2: An Example of Motion Prediction for Vehicles

Motion Prediction. As to vehicles, we adopt the kinematic bicycle model (KBM) [55] to predict the future motions of the EGO vehicle itself and surrounding vehicles. As shown in Fig. 2, KBM is a simplified model of vehicle dynamics, which models vehicles as two-wheel systems, capturing the essential characteristics of vehicle motions, including positions $p = (x, y)$, headings θ, and speeds v. It employs nonlinear dynamics to iteratively compute the future motions of a vehicle over a period of frame in the future using Eq. 2,

$$\frac{d}{dt}\begin{pmatrix} x \\ y \\ \theta \\ v \end{pmatrix} = \begin{pmatrix} v\cos(\theta) \\ v\sin(\theta) \\ v\tan(\delta)/L \\ a \end{pmatrix} \tag{2}$$

where a, δ and L are the acceleration, the steering angle, and the length of the vehicle, respectively.

Specifically, we first obtain EGO vehicle states (*i.e.,* p, v and θ) from standard onboard sensors, and its physical dimensions (*i.e.,* vehicle length L and width W) from vehicle specifications. We parse the a and δ of the EGO vehicle from the trajectory generated by the ADS at frame t. Besides, we utilize the BEV representation BEV^t to get the states of the surrounding vehicles. The BEV^t provides the p, v, θ, L and W of each surrounding vehicles at frame t. We calculate the a and δ of each surrounding vehicle using the difference between their speed and heading in BEV^t and BEV^{t-1}, respectively. Then, we apply the KBM to predict the future motions of the EGO vehicle and surrounding vehicles over a frame duration H, which is set to 60 (*i.e.,* 3 seconds) by default [65]. Finally, we construct the bounding boxes $Box = (x, y, \theta, v, L, W)$ of the EGO vehicle and surrounding vehicles during H by attaching their physical dimensions with predicted motions generated by the KBM. As shown in Fig. 2, following [62], we linearly enlarge each predicted bounding box over frame, up to 130% of the EGO vehicle's original dimensions and up to 200% of surrounding vehicles, to mitigate accumulated prediction errors of KBM.

As to pedestrians, we model the movement of pedestrians with a constant speed assumption along current heading. We obtain the position, speed, heading and physical dimensions of each pedestrian from BEV^t and construct their bounding boxes over the frame period H similar to vehicles, enlarging the size of the original bounding box in BEV^t by 50%.

As to static obstacles, we directly extract the bounding boxes from the BEV representation BEV^t, which are assumed to remain stationary throughout the prediction horizon.

To monitor potential hazards of stop signal violation, inspired by Apollo [5], the influence regions of stop signals (*e.g.,* stop signs and red lights) are abstracted as virtual static bounding boxes. For instance, based on the EGO vehicle's heading, as well as the position and orientation of the stop sign in BEV^t, a 3×3 meter square region is constructed on the lane to represent the stop sign's influence region. The bounding box of the stop sign remains active until the EGO vehicle comes to a complete stop in this region, at which the bounding box is deactivated.

Finally, through motion prediction, we obtain a set of predicted bounding boxes of the EGO vehicle E and all surrounding traffic participants T, including vehicles, pedestrians, static obstacles and stop signals, over the prediction horizon $[t, t+H]$, denoted as $\mathbf{Box}^t = \{Box^i_\alpha \mid \alpha \in E \cup T, i \in [t, t+H]\}$, where each bounding box Box^i_α is denoted as $Box^i_\alpha = (x^i_\alpha, y^i_\alpha, \theta^i_\alpha, v^i_\alpha, L_\alpha, W_\alpha)$. All the predicted bounding boxes are used to evaluate the potential hazards in the next step.

Hazard Evaluation. As mentioned above, we focus on the evaluation of three primary hazards, *i.e.,* the *collision hazard*, the *stop signal hazard*, and the *stalling hazard*.

Collision Hazard. We detect potential collisions by assessing whether the EGO vehicle's predicted bounding boxes Box^t_E result in unsafe or illegal interactions with other traffic participants. Specifically, we identify the intersections using the separating axis theorem [37], which determines $\mathrm{IS}(Box^t_i, Box^t_j) = \mathrm{True}$ if and only if no separating axis exists between Box^t_i and Box^t_j at frame t. With respect to potential collisions, for the $\mathbf{Box}^t$, we calculate the minimum predicted collision frame C^t of the EGO vehicle by Eq. 3.

$$C^t = \min\{i \in [t, t+H] \mid \exists \alpha \in T, \mathrm{IS}(Box^i_{\mathrm{E}}, Box^i_\alpha) = \mathrm{True}\} \tag{3}$$

To reduce the false positive rate, we compare C^t with C^{t-1} that is calculated at previous frame. We consider a potential collision is expected to occur when $C^t \le C^{t-1}$, indicating that the potential collision is becoming more imminent and the ADS trajectory has not shown a tendency to avoid the hazard.

Stop Signal Hazard. A stop signal violation is considered to occur when the EGO vehicle enters the influence region of a stop signal and fails to come to a complete stop (*i.e.,* its speed remains above a predefined threshold $\epsilon = 0.1$ throughout the entire period of intersection with the influence region). For the bounding boxes of the EGO vehicle Box^i_E, the speeds of the EGO vehicle v^i_E, and the bounding boxes of the stop signal Box^i_{sg} over the prediction horizon $[t, t+H]$, we determine the $SigVio = \mathrm{True}$ when Eq. 4 is satisfied.

$$\forall i' \in \{i \in [t, t+H] \mid \mathrm{IS}(Box^i_E, Box^i_{sg}) = \mathrm{True}\}, v^{i'}_E > \epsilon \tag{4}$$

Stalling Hazard. A stalling hazard is considered to occur when the EGO vehicle slows to a near stop (*i.e.,* its speed falls below the predefined threshold ϵ) outside any influence region of stop signals at current frame t. Specifically, for the bounding box of the EGO vehicle Box^t_E and its speed v^t_E at frame t, we determine the $Stalling = \mathit{True}$ when Eq. 5 is satisfied.

$$v^t_E < \epsilon \ \wedge\ \neg\,\mathrm{ValidStop}(Box^t_E, t) \tag{5}$$

Algorithm 1: Waypoint Rerouting

```
Input: EGO vehicle's position: $p_E^t$, EGO vehicle's length: $L$,
       navigation point: $NP$, perception range: $perRange$,
       static obstacles: $SO$, road boundaries: $RB$
Output: the rerouted waypoints: $RW$
1  $ref\_path \leftarrow$ GenerateBézierCurve($p_E^t, NP$);
2  $dense\_wps \leftarrow$ Sampling($ref\_path$);
3  $map \leftarrow$ InitOccupancyMap($p_E^t, perRange$);
4  foreach obstacle or boundary $r \in SO \cup RB$ do
5      $expanded\_r \leftarrow$ ExpandOccupancy($r, L$) ;
6      foreach cell $c \in expanded\_r$ do
7          Mark $c$ as $non\_traversable$ ;
8      end
9  end
10 $RW \leftarrow [p_E^t]$;
11 foreach waypoint $wp$ in $dense\_wps$ do
12     $wp\_cell \leftarrow$ GetCellFromMap($wp$, $map$);
13     if $wp\_cell$ is traversable then
14         Add $wp$ to $RW$;
15     end
16     else
17         $start \leftarrow$ last waypoint in $RW$;
18         $goal \leftarrow$ FindNextTraversableCell($wp$, $dense\_wps$, $map$);
19         $local\_path \leftarrow$ A*($start$, $goal$, $map$) with penalties;
20         Append $local\_path$ to $RW$;
21     end
22 end
23 $RW \leftarrow$ Smooth($RW$);
24 return $RW$;
```

where ValidStop(Box_E^t, t) is a function that checks whether the EGO vehicle is in any influence region of stop signals derived from the BEV representation BEV^t at frame t.

Finally, we maintain the three takeover buffers in the *Takeover Gate*, *i.e.,* the collision buffer $B_{\text{collision}}$ of size M, the stop signal buffer B_{signal} of size M, and the stalling buffer B_{stall} of size N, by Eq. 6,

$$\begin{aligned} B_{\text{collision}}[t \bmod M] &= \mathbb{I}(C^t \leq C^{t-1}) \\ B_{\text{signal}}[t \bmod M] &= \mathbb{I}(SigVio = \text{True}) \\ B_{\text{stall}}[t \bmod N] &= \mathbb{I}(Stalling = \text{True}) \end{aligned} \tag{6}$$

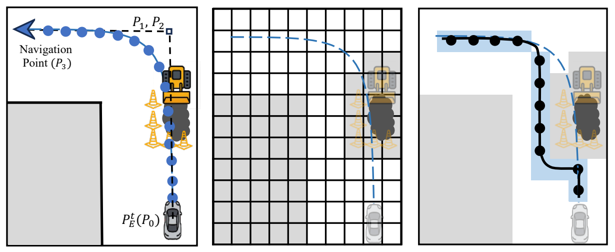
where $\mathbb{I}(\cdot)$ is the indicator function, which returns 1 if the condition is true and 0 otherwise.

During the period of control being taken over, we additionally maintain the recovery buffer B_{recovery} of size R to assess whether the EGO vehicle is safe to return control to the ADS. Specifically, for each frame t, a value of 1 is recorded in the buffer if any potential hazard is detected, otherwise a value of 0 is written, as formulated in Eq. 7.

$$\begin{aligned} B_{\text{recovery}}[t \bmod R] = \mathbb{I}(&B_{\text{collision}}[t \bmod M] \vee \\ &B_{\text{signal}}[t \bmod M] \vee B_{\text{stall}}[t \bmod N]) \end{aligned} \tag{7}$$

D. Hazard Mitigator

To mitigate potential hazards, we adopt a conservative driving strategy based on the IDM, which temporarily takes over the control from the ADS until safety is reestablished. This process involves two key steps, *i.e.,* the **waypoint rerouting**, which generates alternative waypoints to guide the EGO vehicle around obstacles whenever possible, and the **leading actor augmentation**, which dynamically augments the set of leading actors used by IDM to account for different types of hazards, thereby regulating the vehicle speed to ensure safe navigation.



(a) Dense Waypoints (b) Occupancy Map (c) Rerouted Waypoints

Fig. 3: An Example of Waypoint Rerouting

Waypoint Rerouting. At each frame t during the takeover process, we generate a set of guiding waypoints RW for the EGO vehicle based on its current position p_E^t, its length L and the navigation point NP provided by the ADS, and the perception range $perRange$, static obstacles SO along with the road boundary RB extracted from the BEV representation BEV^t. The overall process is illustrated in Algorithm 1, which consists of three main steps, *i.e., Dense Waypoint Generation* (Line 1-2), *Occupancy Map Generation* (Line 3-9), and *Rerouted Waypoint Generation* (Line 10-23). Additionally, Fig. 3 shows an example of the waypoint rerouting, where the gray regions denote non-traversable areas (*i.e.,* road boundaries or static obstacles), and the blue regions show the rerouted waypoints.

Dense Waypoint Generation. First, we construct a reference path ref_path that connects p_E^t and NP (Line 1). Similar to previous work [24, 48], 3-rd Bézier curve [52], which is empirically found sufficient for modeling lane lines, is used to compute a smooth reference path for this phase. A 3-rd Bézier curve $\mathcal{B}$ can be constructed by four control points $P_0 - P_3$, i.e., $\mathcal{B}(\zeta) = (1-\zeta)^3 P_0 + 3(1-\zeta)^2 \zeta P_1 + 3(1-\zeta)\zeta^2 P_2 + \zeta^3 P_3$, $\zeta \in [0, 1]$. Next, we sample points along this reference path to obtain a set of dense waypoints (Line 2). For example, as shown in Fig. 3a, the EGO vehicle is stalled due to a construction zone and the ADS fails to generate a valid trajectory to bypass the obstacle. We set the p_E^t as P_0, and the NP as P_3. Then, we compute the intermediate control point P_1 and P_2, located at the intersection of the heading lines of the EGO vehicle and the navigation point within the drivable area to guide the curve plausibly. Using these four points, we construct a curve and sample it to obtain a set of dense waypoints. Note that the dense waypoints may not always result in a drivable path, especially in the presence of dynamic or complex obstacles. Therefore, the rerouted waypoint generation and the leading actor augmentation work together to ensure safe driving control.

Occupancy Map Generation. After the generation of dense waypoints, we initialize a local occupancy grid map centered on the EGO vehicle. Specifically, an occupancy map is initialized using the current position of the EGO vehicle as the origin and a predefined perception range, covering the surrounding area likely to influence path planning decisions (Line 3). To enable fine-grained path search, we adopt a small grid cell size (e.g., 1.0 m) when generating the occupancy map, which is consistent with practices in robotic navigation [32, 42]. However, unlike robots whose physical footprint is often smaller than a grid cell, vehicles are significantly larger. This discrepancy necessitates additional considerations to ensure safe path planning. We inflate each static obstacle by half the length of the EGO vehicle (Line 5), ensuring no collision occurs between the EGO vehicle and nearby obstacles during waypoint rerouting. Each cell that overlaps with the obstacles or road boundaries is marked as non-traversable (Line 4-9), forming a binary occupancy map that encodes drivable and non-drivable regions for subsequent use in rerouting. For example, as shown in Fig. 3b, the occupancy map is initialized with a perception range of 40 m. For clarity, only a local part of the occupancy map is visualized, where the cells overlapping with the construction zone and road boundary are marked as non-traversable.

Rerouted Waypoint Generation. Using the previously constructed occupancy map, we refine the dense waypoints to safely circumvent obstacles. Starting from the p_E^t (Line 10), we sequentially append each dense waypoint to the rerouted waypoint list RW, unless a waypoint is found to be located in a non-traversable cell of the occupancy map (Line 11-15). The cell corresponding to the last waypoint in the current RW is recorded as a new $start$ point (Line 17), indicating where the original waypoints start to become invalid due to environmental constraints. We use the function *FindNextTraversableCell* to iterate through and remove waypoints that are in non-traversable cells in dense waypoints until it finds a waypoint located in a traversable cell, which is then used as the new $goal$ point (Line 18). Then, from the $start$ point, the remaining portion of the dense waypoints is replanned using an A* [32] search algorithm, initialized with the $start$ as the start node and the $goal$ as the target (Line 19). To encourage safer and more efficient detours, we additionally integrate cost penalties for deviations from the reference dense waypoints and for abrupt directional changes into the A* algorithm. These penalties collectively guide the search process to favor not only the shortest path but also one that conforms to safety and motion feasibility considerations. The resulting path segment is appended to RW, producing a complete, collision-free waypoint sequence (Line 20). After appending the rerouted path segment, the algorithm resumes iterating through the remaining dense waypoints, repeating the process as above. Finally, a smoothing operation is applied to eliminate abrupt turns or jagged transitions before returning the rerouted waypoints, ensuring the continuity and drivability (Line 23-24). Fig. 3c illustrates the rerouted waypoints we generate to navigate around the construction zone.

Leading Actor Augmentation. We utilize the IDM to compute the desired speeds of the EGO vehicle along the rerouted waypoints. Following previous work [70], we set the desired speed v_0 to 0.72 times lane-specific speed limit, s_0 to 4.0m and T to 0.25s; and the maximum acceleration a and the braking deceleration b are fixed to 11m/s^2 and 20m/s^2, respectively. In order to regulate the vehicle speed under potential driving hazards, we augment leading actors for the EGO vehicle, from which we derive the parameters (*i.e.,* the net distance between the EGO vehicle and the leading actor s_E and their relative speed Δv_E) of the IDM introduced in Sec. II-B.

By default, we select the closest vehicle along the rerouted waypoints in front of the EGO vehicle as a leading actor. To further enhance safety, if the *Hazard Monitor* detects a potential collision over the prediction horizon $[t, t+H]$, we additionally include those traffic participants as leading actors, such as vehicles and pedestrians, that are predicted to collide with the EGO vehicle within this horizon. Moreover, static obstacles located along the EGO vehicle's trajectory are also incorporated. These not only help prevent imminent collisions, but also guide the EGO vehicle to decelerate proactively, preserving space for the waypoint rerouting to avoid potential stalling hazards. When stop signals that may affect the EGO vehicle are present, we augment the set of leading actors with virtual static bounding boxes of the stop signals, as designed in Sec. III-C, to restrict the EGO vehicle's speed and ensure it can come to a complete stop within the influence region of stop signals.

Then, we calculate the s_E along with the Δv_E of each leading actor. We apply Eq. 1 to each of these leading actors to generate their respective leading speed, and select the minimum speed as the desired speed to regulate the speed of the EGO vehicle. Finally, the *Hazard Mitigator* sends the newly generated trajectory, consisting of the rerouted waypoints and their desired speed, to the vehicle controller for hazard mitigation, and continues to regulate the vehicle's trajectory until the control is returned to the ADS by the *Takeover Gate*.

E. Formal Guarantees

As defined in Eq. 8, ARGUS guarantees that the system is either in a safe state Φ_{safe} under the control of the *Hazard Mitigator* (HM) at frame t, or remains safe at frame t under the control of the ADS or is able to recover to a safe state at the next frame $t+1$ when the takeover $\mathcal{T}$ is necessary.

$$\begin{aligned}&[\mathcal{C}_t = \text{HM} \wedge s_t \in \Phi_{\text{safe}}] \vee \\ &[\mathcal{C}_t = \text{ADS} \wedge ((\neg\mathcal{T} \wedge s_t \in \Phi_{\text{safe}}) \vee (\mathcal{T} \wedge s_{t+1} \in \Phi_{\text{safe}}))]\end{aligned} \tag{8}$$

where $\mathcal{C}_t$ denotes the ownership of system control at frame t, and s_t denotes the system state at frame t.

This safety guarantee is supported by four key proofs. (a) The initial system state $s_0 \in \Phi_{\text{safe}}$. (b) Given accurate perception and dynamics, our *Hazard Monitor* detects all hazards within our defined hazard set. (c) If $\mathcal{C}_t = \text{HM}$, our IDM-based *Hazard Mitigator* mathematically guarantees no rear-end collision with any leading actors and no stalling, ensuring Φ_{safe} is forward-invariant, *i.e.,* $\forall k \geq 0, s_{t+k} \in \Phi_{\text{safe}}$. (d) If $\mathcal{C}_t = \text{ADS}$ and the takeover is necessary, our deterministic *Takeover Gate* switches control within a latency significantly shorter than a single frame, ensuring $s_{t+1} \in \Phi_{\text{safe}}$ based on (c).

IV. EVALUATION

We have implemented a prototype of ARGUS with 6,709 lines of Python code. The source code and data of our work are available at [2]. To evaluate the effectiveness and efficiency of ARGUS, we design the following six research questions.

- **RQ1 Effectiveness Evaluation**: How effective is ARGUS in enhancing the resilience of end-to-end ADSs?
- **RQ2 Efficiency Evaluation**: How efficient is ARGUS in enhancing the resilience of end-to-end ADSs?

- **RQ3 Accuracy Evaluation**: How accurate is ARGUS in producing appropriate takeover decisions?
- **RQ4 Ablation Study**: How do the steps of *Waypoint Rerouting* and *Leading Actor Augmentation* impact the IDM-based hazard mitigation?
- **RQ5 Parameter Sensitivity Analysis**: How robust is ARGUS in different parameter settings?
- **RQ6 Generalization Evaluation**: How effective is ARGUS when generalized to modular ADSs?

A. Evaluation Setup

Benchmark. To train and evaluate the resilience of the ADS under different driving hazards, we select Bench2Drive [39] and CARLA leaderboard 2.0 validation set [13], which are widely used benchmarks containing various driving hazards. Bench2Drive's training set consists of 2 million fully annotated frames, collected from 12 towns under 23 weathers by the expert model [46] in CARLA [21]. Its evaluation set requires the ADS to pass 44 interactive scenarios under different locations and weathers, which sums up to 220 routes (denoted as Bench2Drive220). Besides, the CARLA leaderboard 2.0 validation set contains 20 long routes with different weather and traffic conditions (denoted as CLV20). The closed-loop driving task requires the ADS to drive toward the destination point under different driving hazards.

Target ADS. We choose three state-of-the-art models (*i.e.,* TCP [75], UniAD [35] and VAD [41]), which are the most representative end-to-end ADSs, as our targets. All the three ADSs are initially trained with the Bench2Drive training set.

Resilience Metrics. First, to measure the overall effectiveness of the ADS under driving hazards, we select the metrics named *Success Rate* (SR), *Route Completion* (RC) and *Driving Score* (DS) [21]. SR measures the proportion of routes that are successfully completed within the allotted time and without any violations. RC_i denotes the completion ratio of route i, while RC represents the average completion ratio across all routes. DS is defined by Eq. 9,

$$DS = \frac{1}{R_{total}} \sum_{i=1}^{R_{total}} RC_i \times IS_i \tag{9}$$

where IS_i is a penalty factor generated by the evaluation benchmark, starting at 1.0, which gets reduced with each violation (*e.g.,* vehicle collision), and R_{total} is the total number of routes.

Second, to evaluate the multiple skills of ADSs under driving hazards, the evaluation routes in Bench2Drive are officially categorized into five groups [39], *i.e.,* merging ($Merge$), overtaking ($Overtake$), emergency braking ($EmgBrake$), giving way ($GiveWay$) and obeying traffic signs ($TSign$), for which success rates are reported per category.

Finally, to evaluate the capability of the ADS to prevent violations in hazardous scenarios, we quantify the number of violations per kilometer from the testing reports provided by CARLA across three categories, *i.e.,* vehicle collision ($Coll$), failure to obey stop signals ($Stop$), and vehicle stalling ($Stall$).

Research Question Setup. For **RQ1**, **RQ2** and **RQ3**, we first bridge the three base models (*i.e.,* TCP, UniAD and VAD) with CARLA, using all routes (*i.e.,* 240 routes in total) in Bench2Drive220 and CLV20. Then, we adopt a widely used approach, *i.e.,* BEVFormer [47], which aggregates multi-view camera inputs to construct the surrounding environment of the ADS. We integrate ARGUS with the three base models based on BEVFormer perception, resulting in TCP-ARGUS, UniAD-ARGUS, and VAD-ARGUS, respectively. In addition, following previous work [14, 25, 46, 72], we use the privileged environment information provided by CARLA, such as surrounding vehicle states and stop signals' location, to construct the BEV representations. We integrate ARGUS with these three base models using the privileged BEVs, and refer to these instantiated variants as TCP-ARGUS*, UniAD-ARGUS* and VAD-ARGUS*, respectively. All the six variants are evaluated on the same set of 240 routes, to assess the effectiveness and efficiency of ARGUS under conditions of realistic perception-based BEVs and the ideal privileged environment information, respectively.

For **RQ1**, we first report the average resilience metrics of the base models and their ARGUS-enhanced variants on two benchmarks. Then, we conduct an in-depth analysis of ARGUS's effectiveness in representative hazardous scenarios.

For **RQ2**, to measure the efficiency of ARGUS, we report the average wall-clock time of each frame in the simulator taken by the three components in ARGUS (*i.e.,* the *Takeover Gate*, the *Hazard Monitor* and the *Hazard Mitigator*), separately.

For **RQ3**, following [63], we leverage the violation timestamps provided by CARLA, and define a takeover occurring within 3 seconds prior to a violation as a necessary intervention. Based on this, we count the number of necessary takeovers as true positives (TP), unnecessary takeovers as false positives (FP), and violations that were not preceded by a takeover as false negatives (FN). Our primary goal is to achieve high recall, defined as Recall = TP / (TP + FN), because the cost associated with false negatives is very high in the safety-critical domain [6], as they may lead to violations. Achieving high precision, defined as Precision = TP / (TP + FP), is also important to avoid unnecessary interventions by ARGUS. Following previous work [63], we report the F$_\beta$ score [6], which is a weighted harmonic mean of precision and recall, with $\beta = 3$ to emphasize recall over precision.

For **RQ4**, we conduct ablation studies to evaluate the individual contribution of the *Waypoint Rerouting* and the *Leading Actor Augmentation* steps within ARGUS. To this end, we implement three reduced variants of ARGUS, *i.e.,* one without *Waypoint Rerouting*, one without *Leading Actor Augmentation*, and one without both steps (*i.e.,* the IDM-only approach). Since evaluating a single ADS using all 240 routes takes at least 5 days, we follow [40] to conduct studies using Dev10 [69], a subset of Bench2Drive220 containing 10 routes, specifically designed to reduce evaluation variance and save computational resources. We evaluate these variants using realistic perception-based BEVs on Dev10 and compare them with their full counterparts (*i.e.,* TCP-ARGUS, UniAD-ARGUS, and VAD-ARGUS) to focus on the average resilience metrics.

For **RQ5**, we conduct parameter sensitivity analysis using realistic perception-based BEVs by varying buffer settings (*i.e.,*

TABLE I: Results of the Overall Effectiveness

Approach	Bench2Drive220			CLV20		
	$SR\uparrow$	$RC\uparrow$	$DS\uparrow$	$SR\uparrow$	$RC\uparrow$	$DS\uparrow$
TCP	14.55	51.53	39.28	0.00	2.21	0.84
TCP-ARGUS	32.73	72.09	60.18	0.00	4.78	1.56
TCP-ARGUS*	38.64	76.02	64.72	0.00	4.81	2.03
UniAD	14.09	68.68	44.62	0.00	2.05	0.50
UniAD-ARGUS	39.55	83.95	68.61	0.00	9.38	2.11
UniAD-ARGUS*	52.27	89.56	75.90	0.00	11.56	2.27
VAD	17.27	61.60	43.31	0.00	0.93	0.37
VAD-ARGUS	39.09	83.66	66.98	0.00	6.15	1.60
VAD-ARGUS*	44.55	85.95	71.51	0.00	6.79	2.09

collision/stalling buffer length M, threshold l, and recovery buffer length R), and bounding box enlargement factors. We evaluate these variants on Dev10 to focus on the average accuracy and resilience metrics.

For **RQ6**, we integrate ARGUS into Apollo 7.0, a widely used open-source modular ADS, and bridge it with CARLA following [29]. We evaluate the effectiveness on Bench2Drive220 and CLV20, comparing ARGUS with REDriver [67], a state-of-the-art safety assurance approach for Apollo by refining trajectories with signal temporal logic to avoid violating traffic rules.

To mitigate the impact of randomness, all the experiments are conducted five times, and the average results are reported.

Environment. We conduct all the experiments on Ubuntu 20.04.4 LTS servers with 4 NVIDIA GeForce RTX 3090 GPUs, Intel(R) Xeon(R) Silver 4310 @ 2.10GHz and 128GB memory.

B. Effectiveness Evaluation (RQ1)

Overall Effectiveness. As shown in Table I, with respect to Bench2Drive220, ARGUS improves SR by 144.00%, RC by 32.65%, and DS by 53.88%, on average, across three ADSs. When the privileged environment information is available, ARGUS* achieves even greater improvements of 198.17% in SR, 39.15% in RC, and 66.66% in DS. With respect to CLV20, we observe that SR remains 0 across all target ADSs and their ARGUS-enhanced variants. This is because the routes in CLV20 are much longer and the hazard scenarios are more complex, making it difficult for the ADSs to fully complete the driving task. Despite this, ARGUS and ARGUS* exhibit improvements in both RC and DS across all target ADSs. Specifically, ARGUS achieves average improvements of 345.05% in RC and 246.72% in DS, while ARGUS* leads to improvements of 403.89% in RC and 320.18% in DS.

To sum up, ARGUS and ARGUS* demonstrate superior improvements across all target ADSs on Bench2Drive220 and CLV20. Compared to ARGUS*, which leverages privileged environment information, ARGUS achieves comparable gains with less reliance on ground-truth data, and its average improvements on SR, RC, and DS are only 4.02, 2.45, 2.91 lower, respectively. These results underscore the value of ARGUS in enhancing resilience, significantly extending the overall effectiveness of target ADSs in the face of hazards.

Multi-Skill Performance. As shown in Table II, ARGUS and ARGUS* significantly enhance the multiple skills of the target ADSs under driving hazards on Bench2Drive220. ARGUS achieves an average improvement of 174.48% on the five driving skills across all target ADSs, with the most significant improvement observed in $TSign$ (333.48%) and the least in $Overtaking$ (22.23%). Besides, ARGUS* achieves an average improvement of 231.57% across all skills, with the most significant improvement observed in $TSign$ (370.14%) and the least in $Overtaking$ (157.42%). Overall, even when relying solely on perceived environmental information, ARGUS effectively enhances the driving skills of the three ADSs, demonstrating its broad applicability and strong generalization capability.

TABLE II: Results of the Multi-Skill Performance

Approach	Bench2Drive220				
	$Merge\uparrow$	$Overtake\uparrow$	$EmgBrake\uparrow$	$GiveWay\uparrow$	$TSign\uparrow$
TCP	14.29	20.00	20.00	10.00	5.91
TCP-ARGUS	34.25	20.00	45.00	30.00	41.40
TCP-ARGUS*	36.36	33.33	46.67	40.00	44.21
UniAD	12.66	13.33	20.00	10.00	13.23
UniAD-ARGUS	29.11	22.22	65.00	50.00	50.53
UniAD-ARGUS*	36.25	51.11	76.67	50.00	57.89
VAD	13.51	20.00	25.42	20.00	23.66
VAD-ARGUS	31.25	20.00	63.33	30.00	51.58
VAD-ARGUS*	31.65	44.44	63.33	50.00	53.19

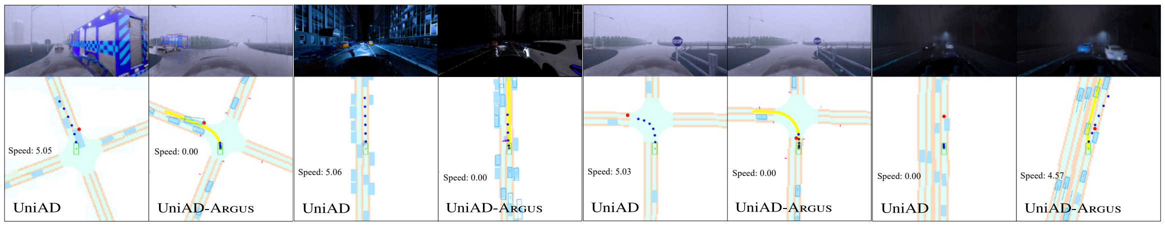
TABLE III: Results of the Violation Prevention

Approach	Bench2Drive220			CLV20		
	$Coll\downarrow$	$Stop\downarrow$	$Stall\downarrow$	$Coll\downarrow$	$Stop\downarrow$	$Stall\downarrow$
TCP	8.23	0.21	3.51	4.64	0.17	2.5
TCP-ARGUS	4.31	0.17	0.64	3.01	0.08	0.75
TCP-ARGUS*	4.19	0.00	0.55	1.65	0.00	0.74
UniAD	10.66	1.51	2.30	11.17	2.46	3.32
UniAD-ARGUS	4.90	0.20	0.61	2.33	0.22	0.79
UniAD-ARGUS*	3.90	0.00	0.48	2.21	0.00	0.63
VAD	10.67	0.41	3.11	16.09	0.42	8.47
VAD-ARGUS	5.16	0.26	0.92	2.59	0.20	1.22
VAD-ARGUS*	4.14	0.00	0.44	2.22	0.00	1.05

Violation Prevention. As shown in Table III, both ARGUS and ARGUS* substantially prevent violations across all target ADSs on Bench2Drive220 and CLV20. On average across all target ADSs on Bench2Drive220, ARGUS reduces $Coll$, $Stop$, and $Stall$ by 51.10%, 51.44% and 75.22%, respectively, while ARGUS* further improves these reductions to 57.06%, 100.00% and 83.10%, respectively. On CLV20, ARGUS reduces $Coll$, $Stop$, and $Stall$ by 66.06%, 65.46%, and 77.02%, while ARGUS* achieves 76.95%, 100.00%, and 79.48%, respectively. ARGUS achieves comparable gains to ARGUS* on the two benchmarks in preventing violations in hazardous scenarios. Notably, ARGUS* achieves perfect compliance with all stop signals under the privileged environment information, fully eliminating the failure to obey stop signals.

Case Study. Fig. 4 illustrates the effectiveness of ARGUS on UniAD in four representative driving hazards, whose videos are available at our replication website [2].

- In Fig. 4a, the EGO vehicle encounters a large emergency vehicle running a red light at a high speed under rainy and low-visibility conditions. UniAD fails to react appropriately and proceeds at a speed of 5.05 m/s, resulting in a collision. In contrast, UniAD-ARGUS correctly monitors the hazard and takes over in time, bringing the vehicle to a full stop (a speed of 0.00 m/s) and thereby avoiding the collision.
- In Fig. 4b, the EGO vehicle encounters a pedestrian emerging from behind a parked vehicle and entering the lane at night. UniAD fails to generate a safe trajectory and adjust its speed in time, leading to a collision with the pedestrian at a speed of



(a) Vehicle Collision (b) Pedestrian Collision (c) Stop Signal Violation (d) Vehicle Stalling

🟩 EGO Vehicle ● Navigation Point ● ADS Trajectory ⌣ Rerouted Waypoints □ Perceived Pedestrians □ Perceived Vehicles ▬ Perceived Stop Signs ▨ Ground-truth BEV

Fig. 4: Qualitative Examples of the UniAD-ARGUS

TABLE IV: Results of Efficiency of Each Module (ms)

Approach	*Takeover Gate*	*Hazard Monitor*	*Hazard Mitigator*
TCP-ARGUS	3e-03	363.45	12.24
TCP-ARGUS*	3e-03	3.94	12.80
UniAD-ARGUS	3e-03	364.43	12.19
UniAD-ARGUS*	3e-03	3.78	12.16
VAD-ARGUS	3e-03	368.31	12.13
VAD-ARGUS*	3e-03	3.82	12.24

TABLE V: Accuracy of the *Hazard Monitor*

Approach	ARGUS			ARGUS*		
	Precision	Recall	F_3	Precision	Recall	F_3
TCP	0.639	0.964	0.917	0.652	0.976	0.930
UniAD	0.664	0.949	0.910	0.710	0.957	0.925
VAD	0.615	0.912	0.870	0.710	0.978	0.943

5.06 m/s. In contrast, UniAD-ARGUS successfully monitors the hazard, takes over control, and stops the vehicle before reaching the pedestrian, thereby avoiding the collision.

- In Fig. 4c, the EGO vehicle approaches an intersection controlled by a clearly visible stop sign. UniAD fails to perform the mandatory stop and continues through the intersection at a speed of 5.03 m/s, resulting in a stop sign violation, while UniAD-ARGUS monitors the potential hazard and takes over to enforce adherence to traffic rules.
- In Fig. 4d, the EGO vehicle encounters a parked vehicle partially obstructing the lane, and must change lanes into same-direction traffic. UniAD fails to plan a feasible trajectory and stalls indefinitely (a speed of 0.00 m/s), while UniAD-ARGUS takes over, reroutes around the obstruction, and safely guides the lane change. Control is then smoothly returned to the ADS, allowing driving to resume without interruption.

Summary. ARGUS can effectively enhance the resilience of end-to-end ADSs in the face of driving hazards, significantly improving their overall effectiveness, multiple driving skills, and the capability to prevent violations.

C. *Efficiency Evaluation (RQ2)*

Table IV reports the average wall-clock time of each frame in the simulator taken by the three components in ARGUS. With respect to the *Takeover Gate* in ARGUS and ARGUS*, the average costs across all three target ADSs are both 3e-03 ms, which is negligible compared to the overall runtime of the target ADS. This is because the *Takeover Gate* is integrated into the trajectory flow of the target ADS as a gating mechanism, which merely checks the takeover and recovery buffers, thereby introducing no measurable runtime overhead to the target ADS.

With respect to the *Hazard Monitor* in ARGUS, the average cost across all three target ADSs is 365.40 ms. This latency is primarily attributed to the use of BEVFormer for aggregating multi-view camera inputs to construct the surrounding environment. However, the *Hazard Monitor* operates in parallel with the target ADS and computes faster than the target ADS's inference speed (*e.g.,* average 534.61 ms per frame for UniAD), ensuring that every predicted trajectory can be monitored without affecting the target ADS's real-time performance. When using the privileged environment information, the average latency of the *Hazard Monitor* in ARGUS* is reduced to only 3.85 ms, proving that the additional cost of the *Hazard Monitor* is primarily due to the BEV representation construction, which can be further optimized in future work.

With respect to the *Hazard Mitigator* across all target ADSs, ARGUS consumes a comparable average runtime of 12.18 ms to ARGUS*, which consumes an average of 12.40 ms. Given that typical end-to-end ADSs require several hundred milliseconds per inference, this additional cost accounts for only approximately 5% average overhead across the three target ADSs, which is practical for mitigating driving hazards in real time.

Summary. ARGUS can efficiently enhance the resilience of end-to-end ADSs and each component in ARGUS incurs little additional time overhead.

D. *Accuracy Evaluation (RQ3)*

Table V presents the accuracy of ARGUS and ARGUS* in producing appropriate takeover decisions. For ARGUS, the average precision, recall, and F_3 score across all three target ADSs are, on average, 0.639, 0.942, and 0.899, respectively. When the privileged environment information is utilized by ARGUS*, these metrics improve further to 0.691 in precision, 0.970 in recall, and 0.932 in F_3 score, on average. Due to the imprecision of the BEV representation generated by the BEVFormer, the *Hazard Monitor* in ARGUS tends to miss some necessary takeovers, while producing more false alarms than that in ARGUS*. Nevertheless, its high recall and F_3 score indicate that ARGUS can still effectively identify the majority of hazardous situations and produce appropriate takeover decisions.

Summary. ARGUS can produce appropriate takeover decisions in the face of driving hazards.

TABLE VI: Ablation Study for the *Hazard Mitigator*

Approach	Step		Overall			Violations		
	Waypoint Rerouting	*Leading Actor Augmentation*	*SR* ↑	*RC* ↑	*DS* ↑	*Coll* ↓	*Stop* ↓	*Stall* ↓
TCP-Argus	IDM-only approach		10.00	78.62	55.38	5.62	0.00	1.12
	✓		10.00	81.63	57.14	4.34	0.00	0.00
		✓	20.00	80.93	64.61	3.30	0.00	1.10
	✓	✓	30.00	90.61	74.87	1.00	0.00	0.00
UniAD-Argus	IDM-only approach		10.00	88.38	52.86	12.32	1.03	1.03
	✓		20.00	91.58	59.33	8.93	0.99	0.00
		✓	50.00	83.70	77.42	0.00	0.00	1.07
	✓	✓	60.00	98.67	83.61	0.00	0.00	0.00
VAD-Argus	IDM-only approach		20.00	74.15	50.99	8.20	1.17	2.34
	✓		20.00	78.67	53.43	7.77	1.11	0.00
		✓	50.00	72.64	69.64	0.00	0.00	1.19
	✓	✓	60.00	92.91	83.91	0.00	0.00	0.00

TABLE VII: Results of Parameter Sensitivity Analysis

(a) Buffer Settings

M	*l*	*R*	Accuracy			Overall		
			Precision	Recall	F_3	*SR* ↑	*RC* ↑	*DS* ↑
1	1	20	0.300	1.000	0.811	30.00	96.41	78.49
3	2	20	0.400	1.000	0.870	40.00	92.64	72.37
5	3	20	0.500	1.000	0.909	40.00	97.96	80.63
5	5	20	0.571	0.667	0.656	20.00	81.35	68.72
10	8	20	0.600	0.500	0.508	20.00	80.11	64.55
5	4	10	0.500	1.000	0.909	40.00	96.85	75.15
5	4	40	0.667	1.000	0.952	50.00	97.29	79.73
5	4	20	0.667	1.000	0.952	60.00	98.67	83.61

(b) Bounding Box Enlargement Factors

EGO Vehicle	Surrounding Vehicles	Pedestrians	Accuracy			Overall		
			Precision	Recall	F_3	*SR* ↑	*RC* ↑	*DS* ↑
100%	100%	100%	0.667	0.667	0.667	40.00	94.33	77.48
160%	300%	200%	0.429	1.000	0.882	20.00	83.72	63.93
130%	200%	150%	0.667	1.000	0.952	60.00	98.67	83.61

TABLE VIII: Results of the Generalization Evaluation

Approach	Bench2Drive220			CLV20		
	SR ↑	*RC* ↑	*DS* ↑	*SR* ↑	*RC* ↑	*DS* ↑
Apollo	12.73	47.61	37.16	0.00	2.12	0.74
Apollo-REDriver	30.45	66.32	52.44	0.00	2.97	1.53
Apollo-Argus	36.81	71.30	60.56	0.00	4.52	1.96

E. Ablation Study (RQ4)

Table VI presents the results of the ablation study for the *Hazard Mitigator*. All reported improvements are based on comparisons with the IDM-only approach. Across all target ADSs, the *Waypoint Rerouting* and the *Leading Actor Augmentation* individually contribute significantly to enhancing the resilience of ADSs, improving DS by 6.73% and 33.23%, respectively, compared with the IDM-only approach. On one hand, the *Waypoint Rerouting* step generates waypoints to avoid static obstacles, resulting in a 4.51% increase, on average, in RC and a complete elimination of $Stall$ violations. On the other hand, the *Leading Actor Augmentation* step introduces augmented leading vehicles to guide speed planning, leading to an 80.43% reduction, on average, in $Coll$ violations and a complete elimination of $Stop$ violations. Besides, when both the two steps are enabled, Argus achieves great improvements in overall effectiveness (*i.e.,* a 300% increase in SR, a 17.40% increase in RC, and a 52.64% increase in DS) as well as in violation prevention (*i.e.,* a 100% reduction in $Coll$ and $Stop$ violations, and a 94.07% reduction in $Stall$ violations).

Summary. Both the two steps contribute the hazard mitigation, and their combination effectively enhances the resilience of end-to-end ADSs, improving the driving performance.

F. Parameter Sensitivity Analysis (RQ5)

Table VII presents the results of our parameter sensitivity analysis using UniAD-Argus. Results of other ADSs are provided at our replication website [2] due to space limitation. For buffer settings, smaller buffers yield consistently higher recall but lower precision, limiting the overall DS metric. Our adopted setting (*i.e.,* M=5, l=4, and R=20) achieves a balanced trade-off with the best overall performance. For bounding box enlargement factors, moderate enlargement ratios outperform both overly conservative and aggressive settings.

Summary. Argus maintains improved resilience across different parameter settings, though overly aggressive or conservative settings cause false positives or missed hazards.

G. Generalization Evaluation (RQ6)

Table VIII presents the overall effectiveness of Argus when generalized to the modular ADS, *i.e.,* Apollo. Compared to the original version of Apollo, Apollo-Argus improves the SR, RC, and DS, on average, by 94.58%, 81.48%, and 113.92% on these two benchmarks, respectively, achieving additional gains of 10.44%, 19.85%, and 21.79% over Apollo-REDriver.

Summary. Argus shows a strong generalization capability to modular ADSs, consistently outperforming both the original Apollo and Apollo-REDriver across different benchmarks.

V. Threats to Validity

First, the quality of BEV representations poses a potential threat to validity. To mitigate this threat, we adopt BEV representations generated by BEVFormer to demonstrate the practical feasibility of our approach under realistic perception conditions. In addition, we leverage privileged BEV representations derived from simulator-provided ground-truth information to validate the upper-bound effectiveness of our framework. We believe that as BEV perception techniques continue to improve, the real-world applicability of Argus will be further enhanced.

Second, the diversity of driving hazards presents another threat. Our current implementation focuses on three critical hazards, *i.e.,* the collision hazard, the stop signal hazard, and the stalling hazard. Our modular design and extensible architecture allow Argus to be expanded to support additional hazards, such as making illegal U-turns in no-U-turn zones, or entering time-restricted entry zones outside permitted hours, where access is controlled based on a predefined schedule (*e.g.,* school zones or peak-hour bus lanes). While these categories cover a wide range of common safety violations, other hazards, such as sensor faults or control delays, lie beyond the scope of our current design and would require separate modeling approaches.

Third, the selection of benchmarks and target ADSs introduces a potential threat to external validity. To mitigate this threat, we evaluate ARGUS on two widely used benchmarks and three representative end-to-end ADSs, capturing the diversity of driving environments and ADS architectures. We conduct a paired t-test analysis [9] across the two benchmarks with diverse towns and weather conditions. ARGUS significantly improves resilience with p-values of 4e-10 and 3e-4 ($p<0.05$) over various towns and weather conditions, respectively. Besides, ARGUS and ARGUS* achieve average p-values of 0.026 and 0.020 on resilience metrics over all target ADSs, respectively, indicating statistically significant improvements ($p<0.05$). These results demonstrate the robustness and general applicability of ARGUS across diverse ADSs and driving environments. Therefore, we believe that ARGUS can be effectively adapted to other end-to-end ADSs to achieve similar improvements. Moreover, since the driving hazards addressed in our work are common in real-world traffic, we believe that ARGUS has strong potential to enhance resilience in real-world ADSs.

Finally, the selection of metrics introduces a potential threat to measuring software resilience. To mitigate this threat, we define software resilience as the ability to adapt to and recover from unexpected events while maintaining effective operation under hazardous conditions. Accordingly, we select a broad set of evaluation metrics. Specifically, we assess resilience in terms of hazard adaptation (*i.e.,* success rate, route completion, driving score, violation prevention, and takeover accuracy) and recovery efficiency (*i.e.,* negligible switching latency).

VI. Related Work

A. Resilience in Autonomous Systems

Resilience in autonomous systems [4, 20, 51] has been widely studied, aiming to enable systems to adapt to and recover from various hazards. Yang et al. [78] enhance the fault tolerance of general-purpose graphics processing unit by selectively replicating thread warps vulnerable to transient faults. MAVFI [34] employs a two-layer autoencoder to monitor the velocity and the time to collision of unmanned aerial vehicles, and triggers the signal to halt the propagation of errors.

In the domain of ADSs, Xia et al. [76] propose a long-range perception system that is robust against sensor misalignment. Meanwhile, Waymo [74] enhances system resilience by deploying redundant hardware that supports immediate failover to maintain safe operation during faults. However, a recent work [71] highlights that evaluating resilience in isolated components of the compute or control stack lacks a holistic, cross-stack perspective, potentially overlooking error propagation and limiting the effectiveness of resilience solutions. To the best of our knowledge, there is no existing work focusing on enhancing the resilience of end-to-end ADSs from a system-level perspective. In this work, we enhance the resilience of end-to-end ADSs at a system level by enabling continuous hazard monitoring and proactive mitigation.

B. Safety Assurance in Autonomous Driving Systems

Ensuring that ADSs operate within strict safety bounds is essential for safeguarding both passengers and other road users, particularly in hazardous scenarios [30, 31, 73]. To monitor driving hazards, prior studies [36, 61, 63, 64] have explored predicting unexpected driving conditions based on distributional shifts or low model confidence. However, such approaches are often impractical for real-time, resource-constrained environments [27]. Alternatively, some works adopt rule-based misbehavior detection [11, 12, 57, 58, 80], which leverage vehicle dynamics to identify potential hazards before safety violations occur. However, such vehicle dynamics-based approaches often depend on accurate physical parameters (e.g., road friction, vehicle mass) and require more computation than the KBM we adopt. Besides, these works primarily focus on detecting hazards and do not provide solutions for adaptive responses to ensure safety. In contrast, our work presents a runtime framework incorporating both continuous hazard monitoring and adaptive hazard mitigation to ensure safety.

Beyond hazard monitoring, some works [66, 67] directly intervene the planning process by modifying each trajectory to enforce specification compliance under the assumption of perfect perception. However, these approaches are tightly coupled with a specific system named Apollo, a modular pipeline ADS, and are difficult to generalize to end-to-end ADSs. Differently, our work is a general approach for end-to-end ADSs. Furthermore, various fallback strategies have been proposed to ensure driving safety under hazardous conditions [17, 28, 36, 77, 79]. Simplex-Drive [16] and RTA-IR [54] focus on avoiding vehicle collisions under *highway-env* [45], without considering the real perception of the sensor. Dual-AEB [82] leverages large language models to trigger appropriate braking responses, but ultimately relies on human-initiated takeovers [26, 81]. Xue et al. [77] further explore fallback strategies that aim to park the vehicle in safe areas during emergencies. However, these approaches primarily emphasize emergency intervention without addressing continuous safety assurance through hazardous scenarios. In contrast, we propose a framework to enhance the resilience of end-to-end ADSs, particularly the ability to continuously monitor hazards, adaptively respond to potential safety violations, and recover quickly to sustain safe operation in hazardous scenarios.

VII. Conclusion

We have proposed and implemented a resilience-oriented runtime framework, ARGUS, to mitigate the driving hazards, thus preventing potential safety violations and improving the driving performance of end-to-end ADSs. Large-scale experiments have been conducted to demonstrate the effectiveness and efficiency of ARGUS.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Grant No. 62332005 and 62372114).

References

[1] S. V. Albrecht, C. Brewitt, J. Wilhelm, B. Gyevnar, F. Eiras, M. Dobre, and S. Ramamoorthy, "Interpretable goal-based prediction and planning for autonomous driving," in *Proceedings of the 2021 IEEE International Conference on Robotics and Automation*, 2021, pp. 1043–1049.

[2] Argus. (2025) Arugs. [Online]. Available: https://argus4ads.github.io/Argus/

[3] D. Atherton, "Incident 434: Sudden braking by tesla allegedly on self-driving mode caused multi-car pileup in tunnel," in *AI Incident Database*, K. Lam, Ed. Responsible AI Collaborative, 2022. [Online]. Available: https://incidentdatabase.ai/cite/434/

[4] S. Bagchi, V. Aggarwal, S. Chaterji, F. Douglis, A. El Gamal, J. Han, B. J. Henz, H. Hoffmann, S. Jana, M. Kulkarni *et al.*, "Vision paper: Grand challenges in resilience: Autonomous system resilience through design and runtime measures," *IEEE Open Journal of the Computer Society*, vol. 1, pp. 155–172, 2020.

[5] Baidu. (2022) Apollo: An open autonomous driving platform. [Online]. Available: https://github.com/ApolloAuto/apollo

[6] D. C. Blair, "Information retrieval, cj van rijsbergen. london: Butterworths; 1979: 208 pp. price: $32.50," *Journal of the American Society for Information Science*, vol. 30, pp. 374–375, 1979.

[7] D. Bogdoll, M. Nitsche, and J. M. Zöllner, "Anomaly detection in autonomous driving: A survey," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 4488–4499.

[8] R. P. Borase, D. Maghade, S. Sondkar, and S. Pawar, "A review of pid control, tuning methods and applications," *International Journal of Dynamics and Control*, vol. 9, pp. 818–827, 2021.

[9] J. F. Box, "Guinness, gosset, fisher, and small samples," *Statistical science*, pp. 45–52, 1987.

[10] J. Cámara, R. De Lemos, N. Laranjeiro, R. Ventura, and M. Vieira, "Robustness-driven resilience evaluation of self-adaptive software systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, pp. 50–64, 2015.

[11] E. Candela, O. Doustaly, L. Parada, F. Feng, Y. Demiris, and P. Angeloudis, "Risk-aware controller for autonomous vehicles using model-based collision prediction and reinforcement learning," *Artificial Intelligence*, vol. 320, p. 103923, 2023.

[12] E. Candela, Y. Feng, D. Mead, Y. Demiris, and P. Angeloudis, "Fast collision prediction for autonomous vehicles using a stochastic dynamics model," in *Proceedings of the 2021 IEEE International Intelligent Transportation Systems Conference*. IEEE, 2021, pp. 211–216.

[13] CARLA. (2025) Carla ad leaderboard 2.1permalink. [Online]. Available: https://leaderboard.carla.org/

[14] D. Chen, V. Koltun, and P. Krähenbühl, "Learning to drive from a world on rails," in *Proceedings of the International Conference on Computer Vision*, 2021, pp. 15 570–15 579.

[15] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li, "End-to-end autonomous driving: Challenges and frontiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[16] S. Chen, Y. Sun, D. Li, Q. Wang, Q. Hao, and J. Sifakis, "Runtime safety assurance for learning-enabled control of autonomous driving vehicles," in *Proceedings of the 2022 International Conference on Robotics and Automation*, 2022, pp. 8978–8984.

[17] K. Cheng, Y. Zhou, B. Chen, R. Wang, Y. Bai, and Y. Liu, "Guardauto: A decentralized runtime protection system for autonomous driving," *IEEE Transactions on Computers*, vol. 70, pp. 1569–1581, 2020.

[18] M. Cheng, Y. Zhou, and X. Xie, "Behavexplor: Behavior diversity guided testing for autonomous driving systems," in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2023, pp. 488–500.

[19] W. M. D. Chia, S. L. Keoh, C. Goh, and C. Johnson, "Risk assessment methodologies for autonomous driving: A survey," *IEEE transactions on intelligent transportation systems*, vol. 23, pp. 16 923–16 939, 2022.

[20] A. Desai, S. Ghosh, S. A. Seshia, N. Shankar, and A. Tiwari, "Soter: a runtime assurance framework for programming safe robotics systems," in *Proceedings of the 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2019, pp. 138–150.

[21] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[22] Y. Fang, H. Min, X. Wu, X. Zhao, X. Wang, G. Li, and R. Teixeira, "Resilience-oriented oscillation dampening for connected autonomous vehicle platoons," *IEEE Transactions on Vehicular Technology*, vol. 73, pp. 16 024–16 040, 2024.

[23] Z. Z. Farah Master and Q. Li, "Xiaomi will cooperate with investigation into fatal ev crash, says founder," in *Reuters*, J. F. Sonali Paul and J. Harvey, Eds. Reuters, 2025. [Online]. Available: https://www.reuters.com/world/china/chinas-xiaomi-says-actively-cooperating-with-police-after-fatal-accident-2025-04-01/

[24] Z. Feng, S. Guo, X. Tan, K. Xu, M. Wang, and L. Ma, "Rethinking efficient lane detection via curve modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 17 062–17 070.

[25] Z. Gao, Y. Mu, C. Chen, J. Duan, P. Luo, Y. Lu, and S. E. Li, "Enhance sample efficiency and robustness of end-to-end urban autonomous driving via semantic masked world model," *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[26] C. Gold, D. Damböck, L. Lorenz, and K. Bengler, ""take over!" how long does it take to get the driver back into the loop?" in *Proceedings of the human factors and ergonomics society annual meeting*, 2013, pp. 1938–1942.

[27] R. Grewal, P. Tonella, and A. Stocco, "Predicting safety misbehaviours in autonomous driving systems using uncertainty quantification," in *Proceedings of the 2024 IEEE Conference on Software Testing, Verification and Validation*, 2024, pp. 70–81.

[28] J. Grieser, M. Zhang, T. Warnecke, and A. Rausch, "Assuring the safety of end-to-end learning-based autonomous driving through runtime monitoring," in *Proceedings of the 23rd Euromicro Conference on Digital System Design*, 2020, pp. 476–483.

[29] guardstrikelab. (2025) Carla apollo bridge. [Online]. Available: https://github.com/guardstrikelab/carla_apollo_bridge

[30] M. Gyllenhammar, M. Brännström, R. Johansson, F. Sandblom, S. Ursing, and F. Warg, "Minimal risk condition for safety assurance of automated driving systems," in *Proceedings of the CARS 2021 6th international workshop on critical automotive applications: robustness & safety*, 2021.

[31] M. Gyllenhammar, G. R. de Campos, and M. Törngren, "The road to safe automated driving systems: A review of methods providing safety evidence," *IEEE Transactions on Intelligent Transportation Systems*, 2025.

[32] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[33] E. Hollnagel, D. D. Woods, and N. Leveson, *Resilience engineering: Concepts and precepts*. Ashgate Publishing, Ltd., 2006.

[34] Y.-S. Hsiao, Z. Wan, T. Jia, R. Ghosal, A. Mahmoud, A. Raychowdhury, D. Brooks, G.-Y. Wei, and V. J. Reddi, "Mavfi: An end-to-end fault analysis framework with anomaly detection and recovery for micro aerial vehicles," in *Proceedings of the Design Automation & Test in Europe Conference & Exhibition*, 2023, pp. 1–6.

[35] Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, W. Wang *et al.*, "Planning-oriented autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 853–17 862.

[36] M. Hussain, N. Ali, and J.-E. Hong, "Deepguard: A framework for safeguarding autonomous driving systems from inconsistent behaviour," *Automated Software Engineering*, vol. 29, p. 1, 2022.

[37] J. Huynh, "Separating axis theorem for oriented bounding boxes," *URL: jkh. me/files/tutorials/Separating% 20Axis% 20Theorem% 20for% 20Oriented% 20Bounding% 20Boxes. pdf*, 2009.

[38] S. International. (2021) J3016: Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. [Online]. Available: https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic

[39] X. Jia, Z. Yang, Q. Li, Z. Zhang, and J. Yan, "Bench2drive: Towards multi-ability benchmarking of closed-loop end-to-end autonomous driving," in *Proceedings of the 38th Advances in Neural Information Processing Systems*, 2024.

[40] X. Jia, J. You, Z. Zhang, and J. Yan, "Drivetransformer: Unified transformer for scalable end-to-end autonomous driving," in *Proceedings of the 13th International Conference on Learning Representations*, 2025.

[41] B. Jiang, S. Chen, Q. Xu, B. Liao, J. Chen, H. Zhou, Q. Zhang, W. Liu, C. Huang, and X. Wang, "Vad: Vectorized scene representation for efficient autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 8340–8350.

[42] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE transactions on robotics*, vol. 21, pp. 354–363, 2005.

[43] D. Krajzewicz, G. Hertkorn, C. Rössel, and P. Wagner, “Sumo (simulation of urban mobility)-an open-source traffic simulation,” in *Proceedings of the 4th middle East Symposium on Simulation and Modelling*, 2002, pp. 183–187.

[44] K. D. Kusano and H. C. Gabler, “Safety benefits of forward collision warning, brake assist, and autonomous braking systems in rear-end collisions,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1546–1555, 2012.

[45] E. Leurent. (2018) An environment for autonomous driving decision-making. [Online]. Available: https://github.com/eleurent/highway-env

[46] Q. Li, X. Jia, S. Wang, and J. Yan, “Think2drive: Efficient reinforcement learning by thinking in latent world model for quasi-realistic autonomous driving (in carla-v2),” in *Proceedings of the 18th European Conference on Computer Vision*, 2024, pp. 142–158.

[47] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Q. Yu, and J. Dai, “Bevformer: Learning bird’s-eye-view representation from lidar-camera via spatiotemporal transformers,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 47, pp. 2020–2036, 2025.

[48] R. Liu, Z. Yuan, T. Liu, and Z. Xiong, “End-to-end lane shape prediction with transformers,” in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2021, pp. 3694–3702.

[49] Y. Lu, Y. Tian, Y. Bi, B. Chen, and X. Peng, “Diavio: Llm-empowered diagnosis of safety violations in ads simulation testing,” in *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2024, pp. 376–388.

[50] Y. Lu, Y. Tian, D. Wang, B. Chen, and X. Peng, “Advfuzz: Finding more violations caused by the ego vehicle in simulation testing by adversarial npc vehicles,” *arXiv preprint arXiv:2411.19567*, 2024.

[51] G. Matthews, L. Reinerman-Jones, D. Barber, G. Teo, R. Wohleber, J. Lin, and A. Panganiban, “Resilient autonomous systems: Challenges and solutions,” in *Proceedings of the 2016 Resilience Week*, 2016, pp. 208–213.

[52] M. E. Mortenson, *Mathematics for computer graphics applications*. Industrial Press Inc., 1999.

[53] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, “A survey of motion planning and control techniques for self-driving urban vehicles,” *IEEE Transactions on Intelligent Vehicles*, vol. 1, pp. 33–55, 2016.

[54] Y. Peng, G. Tan, and H. Si, “Rta-ir: A runtime assurance framework for behavior planning based on imitation learning and responsibility-sensitive safety model,” *Expert Systems with Applications*, vol. 232, p. 120824, 2023.

[55] P. Polack, F. Altché, B. d’Andréa Novel, and A. de La Fortelle, “The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?” in *Proceedings of the 2017 IEEE intelligent vehicles symposium*, 2017, pp. 812–818.

[56] D. J. Provan, D. D. Woods, S. W. Dekker, and A. J. Rae, “Safety ii professionals: How resilience engineering can transform safety practice,” *Reliability Engineering & System Safety*, vol. 195, p. 106740, 2020.

[57] Z. Qian, R. Chen, C. Yi, X. Zhai, and B. Chen, “Collision avoidance control for autonomous driving with multiple dynamic obstacles in iov: A prediction-enhanced apf-based approach,” *IEEE Internet of Things Journal*, 2025.

[58] R. Quinonez, J. Giraldo, L. Salazar, E. Bauman, A. Cardenas, and Z. Lin, “{SAVIOR}: Securing autonomous vehicles with robust physical invariants,” in *Proceedings of the 29th USENIX security symposium (USENIX Security 20)*, 2020, pp. 895–912.

[59] A. W. Righi, T. A. Saurin, and P. Wachs, “A systematic literature review of resilience engineering: Research areas and a research agenda proposal,” *Reliability Engineering & System Safety*, vol. 141, pp. 142–152, 2015.

[60] H. Shao, L. Wang, R. Chen, H. Li, and Y. Liu, “Safety-enhanced autonomous driving using interpretable sensor fusion transformer,” in *Proceedings of the Conference on Robot Learning*, 2023, pp. 726–737.

[61] W. Shao, B. Li, W. Yu, J. Xu, and H. Wang, “When is it likely to fail? performance monitor for black-box trajectory prediction model,” *IEEE Transactions on Automation Science and Engineering*, 2024.

[62] C. Sima, K. Renz, K. Chitta, L. Chen, H. Zhang, C. Xie, J. Beißwenger, P. Luo, A. Geiger, and H. Li, “Drivelm: Driving with graph visual question answering,” in *Proceedings of the European Conference on Computer Vision*, 2025, pp. 256–274.

[63] A. Stocco, P. J. Nunes, M. d’Amorim, and P. Tonella, “Thirdeye: Attention maps for safe autonomous driving systems,” in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 2022, pp. 1–12.

[64] A. Stocco, M. Weiss, M. Calzana, and P. Tonella, “Misbehaviour prediction for autonomous driving systems,” in *Proceedings of the ACM/IEEE 42nd international conference on software engineering*, 2020, pp. 359–371.

[65] Y. Sun, C. M. Poskitt, J. Sun, Y. Chen, and Z. Yang, “Lawbreaker: An approach for specifying traffic laws and fuzzing autonomous vehicles,” in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 2022, pp. 1–12.

[66] Y. Sun, C. M. Poskitt, K. Wang, and J. Sun, “Fixdrive: Automatically repairing autonomous vehicle driving behaviour for \$0.08 per violation,” *arXiv preprint arXiv:2502.08260*, 2025.

[67] Y. Sun, C. M. Poskitt, X. Zhang, and J. Sun, “Redriver: Runtime enforcement for autonomous vehicles,” in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, 2024, pp. 1–12.

[68] Y. Sun, X. Wang, Y. Zhang, J. Tang, X. Tang, and J. Yao, “Interpretable end-to-end driving model for implicit scene understanding,” in *Proceedings of the 2023 IEEE 26th International Conference on Intelligent Transportation Systems*, 2023, pp. 2874–2880.

[69] Thinklab-SJTU. (2025) drivetransformer_bench2drive_dev10. [Online]. Available: https://github.com/Thinklab-SJTU/Bench2Drive/blob/main/leaderboard/data/drivetransformer_bench2drive_dev10.xml

[70] M. Treiber, A. Hennecke, and D. Helbing, “Congested traffic states in empirical observations and microscopic simulations,” *Physical review E*, vol. 62, p. 1805, 2000.

[71] Z. Wan, K. Swaminathan, P.-Y. Chen, N. Chandramoorthy, and A. Raychowdhury, “Analyzing and improving resilience and robustness of autonomous systems,” in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, 2022, pp. 1–9.

[72] H. Wang, X. Ye, F. Tao, C. Pan, A. Mallik, B. Yaman, L. Ren, and J. Zhang, “Adawm: Adaptive world model based planning for autonomous driving,” in *Proceedings of the Thirteenth International Conference on Learning Representations*, 2025.

[73] K. Watanabe, E. Kang, C.-W. Lin, and S. Shiraishi, “Runtime monitoring for safety of intelligent vehicles,” in *Proceedings of the 55th annual design automation conference*, 2018, pp. 1–6.

[74] L. Waymo, “Waymo safety report: On the road to fully self-driving,” 2017.

[75] P. Wu, X. Jia, L. Chen, J. Yan, H. Li, and Y. Qiao, “Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline,” in *Proceedings of the Advances in Neural Information Processing Systems*, 2022.

[76] Z.-X. Xia, S. Fadadu, Y. Shi, and L. Foucard, “Robust long-range perception against sensor misalignment in autonomous vehicles,” in *Proceedings of the 2025 IEEE/CVF Winter Conference on Applications of Computer Vision*, 2025, pp. 5761–5770.

[77] W. Xue, B. Yang, T. Kaizuka, and K. Nakano, “A fallback approach for an automated vehicle encountering sensor failure in monitoring environment,” in *Proceedings of the 2018 IEEE Intelligent Vehicles Symposium*, 2018, pp. 1807–1812.

[78] L. Yang, B. Nie, A. Jog, and E. Smirni, “Enabling software resilience in gpgpu applications via partial thread protection,” in *Proceedings of the 43rd International Conference on Software Engineering*, 2021, pp. 1248–1259.

[79] J. Yu and F. Luo, “Fallback strategy for level 4+ automated driving system,” in *Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference*, 2019, pp. 156–162.

[80] W. Yu, C. Zhao, H. Wang, J. Liu, X. Ma, Y. Yang, J. Li, W. Wang, X. Hu, and D. Zhao, “Online legal driving behavior monitoring for self-driving vehicles,” *Nature communications*, vol. 15, p. 408, 2024.

[81] Y. Yu and S. Lee, “Remote driving control with real-time video streaming over wireless networks: Design and evaluation,” *IEEE Access*, vol. 10, pp. 64 920–64 932, 2022.

[82] W. Zhang, P. Li, J. Wang, B. Sun, Q. Jin, G. Bao, S. Rui, Y. Yu, W. Ding, P. Li *et al.*, “Dual-aeb: Synergizing rule-based and multimodal large language models for effective emergency braking,” *arXiv preprint arXiv:2410.08616*, 2024.

[83] X. Zhang and Y. Cai, “Building critical testing scenarios for autonomous driving from real accidents,” in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2023, pp. 462–474.