ACAV: A Framework for Automatic Causality Analysis in Autonomous Vehicle Accident Recordings

Huijia Sun
ShanghaiTech University
China
sunhj2022@shanghaitech.edu.cn

Christopher M. Poskitt
Singapore Management University
Singapore
cposkitt@smu.edu.sg

Yang Sun
Singapore Management University
Singapore
yangsun.2020@phdcs.smu.edu.sg

Jun Sun
Singapore Management University
Singapore
junsun@smu.edu.sg

Yuqi Chen*
ShanghaiTech University
China
chenyq@shanghaitech.edu.cn

ABSTRACT

The rapid progress of autonomous vehicles (AVs) has brought the prospect of a driverless future closer than ever. Recent fatalities, however, have emphasized the importance of safety validation through large-scale testing. Multiple approaches achieve this fully automatically using high-fidelity simulators, i.e., by generating diverse driving scenarios and evaluating autonomous driving systems (ADSs) against different test oracles. While effective at finding violations, these approaches do not identify the decisions and actions that *caused* them—information that is critical for improving the safety of ADSs. To address this challenge, we propose ACAV, an automated framework designed to conduct causality analyses for AV accident recordings in two stages. First, we apply feature extraction schemas based on the messages exchanged between ADS modules, and use a weighted voting method to discard frames of the recording unrelated to the accident. Second, we use safety specifications to identify safety-critical frames and deduce causal events by applying CAT—our causal analysis tool—to a station-time graph. We evaluated ACAV on the Apollo ADS, finding that it can identify five distinct types of causal events in 93.64% of 110 accident recordings generated by an AV testing engine. We further evaluated ACAV on 1206 accident recordings collected from versions of Apollo injected with specific faults, finding that it can correctly identify causal events in 96.44% of the accidents triggered by prediction errors, and 85.73% of the accidents triggered by planning errors.

CCS CONCEPTS

• **Computer systems organization → Embedded and cyber-physical systems**; **Maintainability and maintenance**; • **Software and its engineering**;

KEYWORDS

Autonomous driving system, test reduction, causality

*Yuqi Chen is the corresponding author.

ACM Reference Format:
Huijia Sun, Christopher M. Poskitt, Yang Sun, Jun Sun, and Yuqi Chen. 2024. ACAV: A Framework for Automatic Causality Analysis in Autonomous Vehicle Accident Recordings. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE '24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3597503.3639175

1 INTRODUCTION

Autonomous Vehicles (AVs) are set to bring about a paradigm shift in transportation. AVs operate through the use of advanced Autonomous Driving Systems (ADSs), which eliminate the need for human drivers to control the vehicle's movements. ADSs are considered highly security-critical systems, as malfunctions can result in severe consequences [1, 4, 22]. For example, a minor error in trajectory prediction can lead to potentially hazardous or even fatal situations for passengers, other road users, and pedestrians. Thus, it is imperative for AV developers to subject ADSs to rigorous testing to ensure their accuracy and reliability. Given that on-road testing suffers from several limitations (such as safety risks and high expenses), simulation-based testing in high-fidelity simulators such as SVL [49] and CARLA [23] has emerged as a popular approach for evaluating AVs.

Many researchers utilize search-based [7–9, 24, 47, 64] and sampling techniques [15, 28, 33, 56, 61, 62] to generate and execute test cases against a set of testing oracles in simulation environments. This provides a controlled and repeatable means of evaluating AVs without the risks associated with real-world testing. For instance, AV-Fuzzer [38] uses fuzzing to generate scenarios that cause safety violations such as near- and actual collisions. LawBreaker [52], also based on fuzzing, further evaluates AVs against specifications of national traffic laws (e.g., rules for crossing junctions). While these methods are effective at finding different violations, they typically do not provide insight into the specific decisions and actions of the AV that ultimately *caused* the violations. Such information is critical for engineers to improve the safety and reliability of AVs but is time-consuming and laborious to extract manually, especially in large-scale testing frameworks. This problem has been emphasized in a recent study [39]: given the vast amounts of driving recordings collected during testing, there is an urgent need for automated tools to support ADS engineers, e.g., in tasks such as clipping and interpreting.

Causality analysis has been proposed within the software engineering community as a means to assist developers in deducing the underlying causes of faulty behaviors observed in a failed test case. This technique has shown notable effectiveness in analyzing complex systems [11, 18, 19, 59]. Unfortunately, given an AV accident recording extracted from a simulator, it is non-trivial to apply existing causality analysis techniques due to two main challenges. First, ADSs consist of multiple independent, decoupled modules that communicate via message passing. Thus, minor faults in one module can eventually propagate into serious faults in other modules. For instance, an incorrect trajectory prediction may be used by an AV's planning module in a way that leads to an accident: in this context, the planning module is not solely to blame. Second, the analysis space of a typical accident recording is huge, requiring new approaches for identifying the accident-related segments that should be focused on.

To address these challenges, we present ACAV, a framework for Automatic Causality analysis of AV accident recordings. Our approach consists of two stages: *accident recording simplification* and *causality analysis*, summarized in the high-level workflow diagram of Figure 1. In the first stage, we define and apply feature extraction schemas based on the messages exchanged between ADS modules. These schemas are used to vectorize information about the map, as well as the AV's perception, prediction, and planning. We then propose a weighted voting method to integrate the slicing plans generated by these schemas, allowing for segments unrelated to the safety violation to be discarded. In the second stage, we identify safety-critical frames using an a priori method based on safety specifications extracted from the driver's handbook and traffic laws of California. Next, we apply our novel causality analysis tool, CAT, to identify the causal events of an accident by analyzing the Station-Time graph (ST graph). In conclusion, our framework is designed to identify the safety-critical frames that brought about an accident, and to generate detailed reports enumerating potential causes. This functionality empowers engineers to gain a comprehensive understanding of the accident dynamics without first needing to replay entire recordings.

To evaluate the effectiveness of our framework, we implemented it for Apollo 7.0 [2] and the SVL simulator [49], which are widely used tools in the field of autonomous driving research and development. Using an AV testing engine [52], we collected a total of 110 accident recordings, including accidents involving intersections, merging, and tailgating. We applied ACAV to vectorize and simplify these recordings, finding that ACAV achieved a 62.23% reduction ratio rate without discarding critical frames, demonstrating its effectiveness in simplifying accident recordings. Upon analyzing the simplified recordings with CAT, our approach identifies five distinct types of causal events in 93.64% of the recordings, including incorrect priority prediction (found 26 times), incorrect trajectory prediction (51 times), improper behavioral planning (17 times), unsafe motion planning (67 times), and vehicle out-of-control (103 times). Finally, we further evaluated ACAV on 1206 accident recordings collected from versions of Apollo injected with specific faults, finding that it can correctly identify causal events in 96.44% of the accidents triggered by prediction errors, and 85.73% of the accidents triggered by planning errors.

Our website [6] provides videos of multiple accidents involving the Apollo ADS, together with the complete accident reports generated by ACAV, as well as our source code.

Overall, we make the following contributions:

- Feature extraction schemas for vectorizing map, perception, prediction, and planning information from ADS messages in AV accident recordings.
- A mechanism for identifying and discarding recording segments unrelated to the accident.
- A tool for identifying safety-critical frames from an accident recording by leveraging ST graphs.
- ACAV, which to the best of our knowledge, is the first modular framework for AV accident analysis and explanation.
- An implementation for Apollo 7.0 and SVL that is able to identify five types of causal faults in AV accident recordings.

The paper is organized as follows. In Section 2, we review some essential background and present a motivational example. Section 3 introduces the design of ACAV, including the detailed algorithms of its two stages. Section 4 evaluates whether ACAV achieves its goal of identifying causal events from AV accident recordings. Finally, Section 5 compares our approach against some related work, before Section 6 concludes.

2 BACKGROUND AND EXAMPLE

2.1 Multi-Module ADSs

The ADSs of AVs are composed of various modules, including perception, localization, prediction, planning, and control. These modules utilize multiple sensors, such as cameras, LiDAR, GNSS, and IMU, that capture raw data (e.g., images, 3D point clouds) about the AV's state as well as the environment it is operating in. To facilitate collaboration among the modules, industrial-level ADSs use a publish-subscribe (i.e., message-based) model for communication. Each module subscribes to one or more channels in the ADS to obtain the required inputs and publishes its output as a message to the corresponding channels.

Specifically, the localization module constantly processes data collected from the GPS, IMU, and (sometimes) LiDAR, then publishes messages containing information about the vehicle's position, orientation, and speed. The perception module receives this data, along with additional information from cameras and radars, then publishes data about perceived obstacles in front of the AV. The prediction module receives the messages published by the perception and localization modules to predict the trajectory of the detected obstacles, and publishes the results to the prediction channel. The planning module subscribes to messages from all of the previous modules to make driving decisions, e.g., determining the appropriate speed and acceleration. Finally, the control module converts the trajectory points generated by the planning module into control commands for the chassis, such as steering, throttle, and brake, to ensure the vehicle travels according to the planned trajectory.

Planning module. The ADS's planning module performs three main functions: *route planning*, *behavioral planning*, and *motion planning* [31, 45, 50]. Given a destination, *route planning* selects a route by choosing a list of lanes and junctions from the map. This route serves as the reference line for behavioral planning and motion planning. Behavioral planning is responsible for making

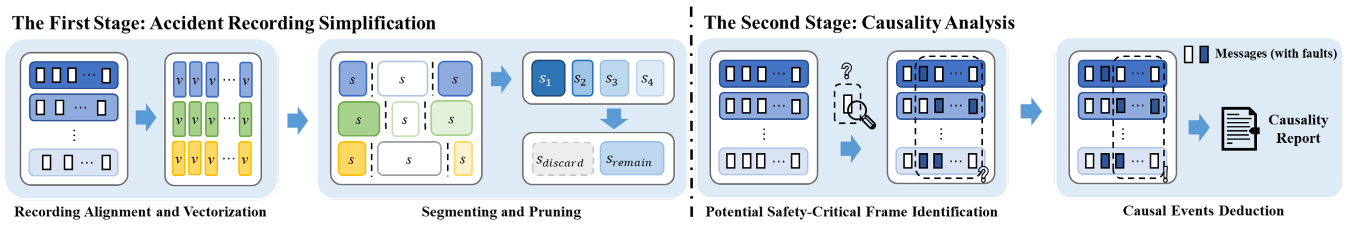

The First Stage: Accident Recording Simplification
The Second Stage: Causality Analysis
Recording Alignment and Vectorization
Segmenting and Pruning
Potential Safety-Critical Frame Identification
Causal Events Deduction
Messages (with faults)
Causality Report
s_discard
s_remain

Figure 1: Overview of ACAV: the first stage vectorizes data exchanged between ADS modules and discards recording segments irrelevant to the accident; the second stage performs a causality analysis using the CAT tool

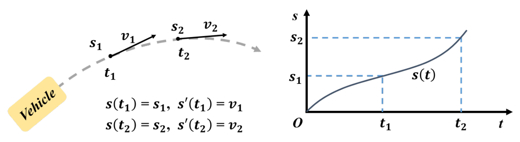

Vehicle
$s(t_1) = s_1,\ s'(t_1) = v_1$
$s(t_2) = s_2,\ s'(t_2) = v_2$
s(t)

Figure 2: A visual example of an ST graph

high-level driving decisions based on the current driving scenario to interact with pedestrians and other vehicles safely. For instance, when the AV detects a construction area ahead of its lane, behavioral planning needs to consider both the dynamic behavior of surrounding traffic participants and the road conditions to decide how to bypass it, e.g., by changing lanes. Lastly, *motion planning* translates high-level decisions into a series of waypoints as part of an executable trajectory, which can be translated into throttles and steering commands by the control module.

Behavioral and motion planning are critical tasks of the planning module, translating the path obtained from route planning into a series of waypoints by calculating specific speed and acceleration plans. This ensures that the AV interacts safely and comfortably with other traffic participants in the current scenario. Various planning techniques employ distinct approaches to integrate the three essential functions. For example, the lattice planner [57], a graph search-based technique, performs behavioral planning and motion planning implicitly and simultaneously under the guidance of well-designed cost functions. In contrast, the EM planner [26] performs behavioral planning and motion planning explicitly and step-by-step. In addition, the Frenet frame method is a well-known approach for describing the motion and trajectories of vehicles, which decouples vehicles' lateral and longitudinal motion, corresponding to the lateral and longitudinal control. The longitudinal behavioral and motion planning can be visualized effectively in a Station-Time graph (ST graph), where time is the horizontal axis, the planned longitudinal trajectory distance is the vertical axis and the planned longitudinal trajectory is a curve, as shown in Figure 2. Additionally, the curve's gradient represents the longitudinal speed of the vehicle.

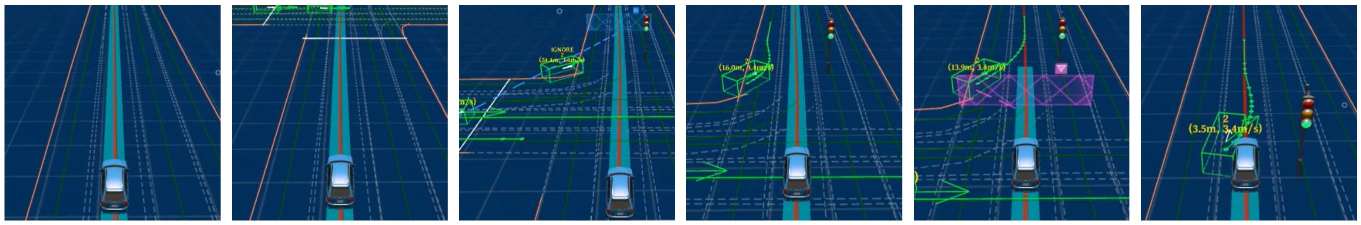
2.2 Motivating Example

To introduce the concept of accident causality and demonstrate how our framework works, we elaborate with an accident driving recording collected from version 7.0 of Apollo. As illustrated in Figure 3, we summarize the scenario in an accident driving recording as six critical scenes, the demo video of which can be found on page 3 of 'Video Demos' on [6]. Initially, the AV drove alone without encountering any traffic signals (Scene 3a). However, it later detected traffic signals and non-player characters (NPCs) as it was approaching an intersection (Scene 3b). The AV made an 'overtake' decision with respect to NPC 4 and executed it (Scenes 3c–3d). After overtaking NPC 4, it interacted with NPC 2, making a 'yield' decision, but still collided with NPC 2 (Scenes 3e–3f).

Accident-related recording segment. As we described above, the AV did not detect any NPCs nor was it near any NPCs in Scenes 3a–3b. In contrast to the other four scenes, these first two had no impact on the accident. Furthermore, during simulation tests, the AV persisted in moving forward *after* a collision, rather than stopping; this behavior, too, had no influence on the accident. Given that our objective is to perform accident analysis, our framework is designed to automatically identify and exclude such segments that are unrelated to the accident. The remaining segments are then fed into the causality analysis stage of our framework. For this recording, ACAV can significantly reduce its length from 17 seconds to 4 seconds, without removing any critical frames. This reduces the workload of ADS engineers who can then immediately focus on the most important parts of the recording.

Safety-critical frames. As depicted in Scene 3c, the AV made an 'overtake' decision with respect to NPC 4 near the intersection. This decision was unsafe because it violates traffic regulations and increases the risk of accidents. Our framework uses a priori knowledge to label frames containing potential accident risks, such as this one, as safety-critical frames. Moreover, each of the frames marked as safety-critical (e.g., Scenes 3c–3e) will be individually inspected by our framework.

Causality analysis. Our framework can automatically identify causal events in an accident recording, as shown in Table 1. According to the table, the AV chose to overtake NPC 4 at the intersection due to failing to predict NPC 2's trajectory (Scenes 3c–3d). When the AV finally correctly predicted the trajectory of NPC 2 (Scene 3e), it was traveling at a speed of 39km/h (approximately 10.83m/s) and was less than 14 meters away from the NPC. Despite making the appropriate 'yield' decision, there was insufficient time and space to carry it out, resulting in a collision. The causal factors of this accident can be attributed to the incorrect priority and trajectory prediction by the prediction module, the flawed decision made by the planning module, and the vehicle's skidding.



(a) (b) (c) (d) (e) (f)

Figure 3: Motivating example: six key scenes from a recording of an AV accident

3 FRAMEWORK DESIGN

As illustrated in Figure 1, our framework consists of two main stages: *accident recording simplification* and *causality analysis*. In the following sections, we provide a detailed explanation of the two stages of our framework and present an implementation of it for Baidu Apollo 7.0. Our framework is available online [6].

3.1 Stage #1: Accident Recording Simplification

In the first stage, the primary objective is to extract short segments from a long driving recording related to an accident. The idea is to provide coarse-grained filtering based on the scenario information. To accomplish this, we partition the driving recording into a series of frames and assign three vectors to each frame to capture information on the current scenario, ranging from the environment to the maneuvers and status of the AV and NPCs. We use a scenario-based recording segmentation technique to merge the frames, and subsequently, we use specifications derived from the driver handbook and traffic laws of California to identify segments of the recording that are relevant to accidents.

Data Collection. A prerequisite for our approach is to collect several accident driving recordings, which requires generating numerous testing scenarios for testing ADSs holistically against different safety oracles. To satisfy this prerequisite, we adapted the AVUnit framework [63], which provides domain-specific languages (DSLs) for specifying testing scenarios and oracles, as well as a fuzzing engine for obtaining effective test cases. Our adaptation extends the fuzzing engine by adding a recorder that captures the corresponding driving recording for each test case, i.e., each test case is captured in a single recording file. The set of initial configurations we used in our experiments includes different combinations of starting points, destinations, and NPCs. To handle the varying routes of all the combinations, the duration of each recording file was set at 60 seconds, which can cover all possible durations of a single test case. We ran the fuzzing engine for two days, generating 1260 test cases, including 131 accident test cases. The combined length of all these recordings exceeded 21 hours. After the termination of the fuzzing algorithm, we selected 110 accident driving recordings based on the output of AVUnit and classified them into three categories (intersection, merging, and rear-end accidents), excluding 21 accident test cases in which the car crash occurred after the AV stopped at its destination or the AV got hit from behind by an NPC.

Table 1: Results of a causality analysis for the example

Time	Accident causal events	Details
0s	AV keeps safe distance from NPCs	–
0.4s	Wrong motion planning; AV skidding sometimes	AV's planning speed is too fast or too slow.
0.8s	Wrong planning caused by the wrong prediction; AV skidding sometimes	For NPC 2: wrong priority prediction. For NPC 4: improper 'overtake' decision. AV's planning speed is too fast or too slow.
2.6s	Wrong motion planning; AV skidding sometimes	AV's planning speed is too fast or too slow.
4.3s	Accident!	

Message Alignment and Vectorization. In a multi-module ADS, the modules collaborate by asynchronously exchanging and processing messages. The content of each message varies depending on the module that published it. To facilitate causality analysis, we select and align messages from the communication channels of the map, localization, perception, prediction, and planning modules, each of which have different publishing frequencies. We divide the recording into several frames, each of which has a duration of 0.08s (chosen because the localization module has the fastest frequency, publishing messages every 0.08s). If these channels publish messages within the frame, we hold the messages and align them to the beginning of the frame. If a channel does not publish any messages within the frame, we copy the last message generated before it to the beginning of the frame.

In the vectorization phase, our primary objective is to extract information related to accidents, which can be associated with the map, perception, prediction, and planning modules in the ADS. To comprehensively capture information from across these modules' messages, we have designed three feature extraction schemas: one for the map, one for perception and prediction, and one for planning. Each schema includes factors that impact the AV's planning, or properties that reflect its current planning status.

The map schema contains information on whether the AV is at a junction, crosswalk, or near a stop sign, as well as the color of the perceived traffic signal. The perception and prediction schema includes four lists of NPCs, indicating which NPCs the AV is approaching, which are in close proximity, and which are predicted as ones to take 'caution' of or 'ignore'. The planning schema includes information on the main driving decision the AV currently executes, the operational design domain (ODD), motion, and whether it is safe according to the responsibility-sensitive safety (RSS) rules [51].

It is worth noting that these are fundamental components among industry-level ADSs such as Autoware and Apollo. Specifically, the ODD defines the specific operating conditions and scenarios in which an AV is designed to function safely and effectively. For instance, Autoware's ODDs include 'Lane Following', 'Lane Change,' and 'Pull Out,' among other scenarios, each suggesting the appropriate scene module in Autoware that should be launched to handle the specific driving situation. Similarly, Apollo's ODDs consist of scenarios such as 'Lane Change,' 'Lane Borrow,' and 'Path Assess,' indicating the corresponding decider/optimizer in Apollo that should be activated to make informed driving decisions. To ensure safety and responsible behavior, the planning schema utilizes RSS rules, which are designed to formalize concepts such as dangerous situations, appropriate responses, and the allocation of blame in a mathematically rigorous manner. Our framework converts each frame into three feature vectors based on these three schemas. Each feature vector contains specific semantic properties, with each dimension representing a particular attribute.

For instance, in the feature vector for the map schema, we have four dimensions indicating whether the AV is: 1. Near an intersection (The distance between the AV and the intersection is less than 5 meters); 2. Near a crosswalk (The distance between the AV and the crosswalk is less than 5 meters); 3. Near a stop sign (The distance between the AV and the stop sign is less than 5 meters); 4. Detected traffic signals. Thus, the vector $\langle False, False, True, None\rangle$ indicates that in the current frame, the AV is approaching a stop sign, not in an area near an intersection or a crosswalk, and not encountering any traffic signals. In this way, we transform the driving recording into a list of feature vectors while preserving the abstract semantic information of each frame, facilitating subsequent segmenting and pruning.

Segmenting and Pruning. After the frame vectorization stage, the framework segments the recording by comparing the similarity of consecutive feature vectors. The idea is to group together sequential frames with identical feature vectors into a single segment. For example, if the AV drives on a road segment for 100 uninterrupted frames, then the feature vectors of these 100 frames are the same, and they will be clustered as a single segment based on the static map environment schema. Our framework generates segmentation plans for each of the three types of vector schemas previously described. These segmentation plans fuse vectors together using a weighted voting method that determines the optimal clipping point. For each frame, a general voting function can be defined for any weighted combination of feature vectors. Let w_c denote the weighted value of c feature vectors, and v_c denote the vote by the c feature vectors. Let

$$voting(v_{map}, v_{perc}, v_{pred}) := \sum_{c\in C} w_c \times v_c \geq \frac{1}{2}\sum_{c\in C} w_c \qquad (1)$$

where $C = \{map, perc, pred\}$, which returns $True$ or $False$, indicating (respectively) whether the current vector should be deemed as a clipping point (i.e., last frame of the segment) or not. The weight of the vote by each category is discussed in Section 4.2.1.

Numerous AV accident reports indicate that most accidents happen in specific contexts, e.g., at intersections, or when there are multiple traffic participants [20, 27, 55]. Armed with this knowledge, our approach creates an overapproximation of relevant frames to narrow down our focus to the most crucial situations. To achieve this, we seek out and discard *irrelevant frames* by analysing static map environments as well as perception and prediction information. To classify a frame as irrelevant, we consider several factors. First, we check if the static environment of the frame includes a junction, a crosswalk, or a stop sign. Next, we verify that the AV is neither approaching nor near any NPC in the frame. Finally, we ensure that the AV does not predict a 'caution' or 'ignore' priority for any NPC. If all of these conditions are met, we classify the frame as an irrelevant frame. To determine whether to discard a segment S, we count the irrelevant frames within it using function $count(S)$, and compute the irrelevant frame ratio $r_m = \frac{count(S)}{len(S)}$. If r_m is larger than the threshold th_m, S will be discarded, otherwise, it will be kept. We discuss the selection of a particular threshold th_m in Section 4.2.1.

Algorithm 1 summarises the steps of our segmenting and pruning method. Specifically, given three categories of feature vectors of an aligned recording, for a feature vector of a frame within it, if the vector is different from its previous one, then we deem that the frame gets one vote by one of the three feature vector categories (Lines 5–10). After collecting votes from all three categories, we perform voting (Line 11) to decide whether to slice in this frame (Line 12–14), the definition of which is shown in Equation 1, and the weight selection of which is discussed in Section 4.2.1. We prune the accident-related segments by examining the segments (working backwards) at Lines 17–26. The last segment is deemed as a part of the accident-related segment (Line 17). For other segments, we find the irrelevant map or perception vectors and determine whether to discard them. For a non-irrelevant segment, we merge it into S_a if S_a follows it, as shown in Lines 21–25. We discuss the selection of a threshold value in Section 4.2.1.

3.2 Stage #2: Causality Analysis

In the second stage, we automatically analyze the accident-related segments that were generated in the first stage to identify potential causes of the accident. We utilize automotive safety specifications from California's driver handbook [16] and traffic laws [17] to identify safety-critical frames that may have contributed to the accident. Next, we implement a causal analysis tool, CAT, that works by examining speed planning. For frames that are identified as suspicious, CAT compares their current speed planning and actual trajectory to deduce the causal events of the accident. This process enables our framework to effectively identify the causes of the accident and provide valuable insights for future improvements.

Potential Safety-Critical Frame Identification. In order to identify safety-critical frames in an accident-related driving recording segment, our framework uses a frame checker that utilizes a priori knowledge, i.e., a list of specifications extracted from background knowledge. In particular, we examine California's driver handbook [16]—published by the Department of Motor Vehicles (DMV)—and traffic laws [17], to obtain a list of specifications for each stage. These specifications include identifying critical obstacles, improper priority prediction, and driving decision-making.

In order to ensure compliance with the rules outlined in the driver's handbook [16], it is necessary to have a robust specification language that allows us to precisely describe these rules. To this end,

Algorithm 1: Segmenting and Pruning

```
Input: $V$: all the three categories of feature vectors of the original
       aligned recording before the accident with length $n$;
Output: $S_a$: the reduced accident-related segment;
$St \leftarrow \emptyset$;
$ss \leftarrow 1$;
$se \leftarrow 0$;
for $i \leftarrow 2$ to $n$ do
    $v_{map} \leftarrow 0$; $v_{perc} \leftarrow 0$; $v_{pln} \leftarrow 0$ ;
    for $c$ in $\{map, perc, pred\}$ do
        if $V_c[i] \neq V_c[i-1]$ or $i == n$ then
            $v_c \leftarrow 1$;
        end
    end
    // See Section. 4.2.1 for voting method
    if voting($v_{map}, v_{perc}, v_{pln}$) then
        $se \leftarrow i$ ;
        $St.push(V[ss : se])$ ;
        $ss \leftarrow i$ ;
    end
end
$S_a \leftarrow St.pop()$ ;
while $St.isNotEmpty()$ do
    $S_{curr} \leftarrow St.pop()$ ;
    $r_m \leftarrow \frac{count(S_{curr})}{len(S_{curr})}$;
    // See Section. 4.2.1 for threshold
    if $r_m \leq th_m$ then
        if $S_a$ is succeeded by $S_{curr}$ then
            $S_a \leftarrow S_{curr} + S_a$;
        end
    end
end
return $S_a$
```

Table 2: State variables in the specification language

Variable	Type	Remarks
$x.spd$	Number	Speed of vehicle x
$x.onJct$	Bool	True if and only if the vehicle x is on a junction
$x.onCswk$	Bool	True if and only if the vehicle x is on a crosswalk
$x.predTraj$	Waypoints	vehicle x's predicted trajectory by the EV
$x.isPed$	Bool	True if and only if the vehicle x is a pedestrian
$x.isBcycl$	Bool	True if and only if the vehicle x is a bicyclist
$x.bhndEV$	Bool	True if and only if the vehicle x is in an area behind the EV
$x.blndEV$	Bool	True if and only if the vehicle x is in the blind area of the EV

we have adopted a specification language based on propositional logic. The specification language consists of propositions (based on a set of pre-defined variables), as well as the usual logical connectors. Before introducing the specifications, we first introduce the pre-defined variables, which can be organized into three categories: state variables, deviation variables, and maneuver variables.

Firstly, the state variables describe the states of vehicles. Table 2 lists a subset of these variables and their usage in describing vehicle properties. For instance, suppose there is an NPC *npc* driving near a junction with a speed of 5m/s to the front-left of the AV, then $npc.spd$ is 5m/s, $npc.onJct$ is $True$, and $npc.predTraj$ contains the

Table 3: Deviation variables in the specification language

Variable	Type	Remarks
Th_{err}	Number	The threshold of the error of trajectory prediction
$MaxBound$	Number	The maximum speed limit of a road segment
$MinBound$	Number	The minimum speed limit of a road segment
$dist(x, y)$	Number	The distance between two objects x and y
$Err(t)$	Number	The error of the trajectory prediction t
$CritObst(x)$	Bool	True if and only if $dist(ev, x) < 3 \times ev.speed$ for an NPC x

Table 4: Maneuver variables in the specification language

Variable	Type	Remarks
$PrioIgn(x)$	Bool	True if and only if the EV predicts NPC x as an "ignore" priority
$DecnIgn(x)$	Bool	True if and only if the EV makes an "ignore" decision on NPC x
$DecnFlw(x)$	Bool	True if and only if the EV makes an "follow" decision on NPC x
$DecnYld(x)$	Bool	True if and only if the EV makes an "yield" decision on NPC x
$DecnOvtk(x)$	Bool	True if and only if the EV makes an "overtake" decision on NPC x

waypoints in the predicted trajectory of *npc*. The other variable values of type *bool* are all $False$.

Secondly, Table 3 summarizes deviation variables to specify various deviation calculations. Here, $MaxBound$ and $MinBound$ represent the upper and lower speed limits of the road on which the AV is traveling. Functions $dist(x, y)$ and $Err(t)$ represent (respectively) the distance between two objects and the error in trajectory prediction. Additionally, we define the function $CritObst(x)$ to filter out the NPCs that need to be focused on in a given scenario. The function $CritObst(x)$ outputs $True$ if and only if the distance between object x and the AV is less than three times the current speed of the AV.

Finally, the (subset of) maneuver variables presented in Table 4 reflect the prediction and planning status of the AV. These variables are directly extracted from prediction and planning messages. For example, if the AV is closely and cautiously following an NPC *npc*, then $PrioIgn(npc)$ would be $False$, and $DecnFlw(npc)$ would be $True$. The remaining maneuver variables would be set to $False$. Here, the AV's priority prediction for an NPC can be roughly divided into three types: 'caution' for a critical NPC, 'ignore' for an immaterial NPC, and 'normal' for the rest. The AV's driving decision towards an NPC can be summarized as a list of maneuvers, including 'ignore', 'stop', 'follow', 'yield', 'overtake', 'nudge', etc.

With the defined variables, we can now describe the specifications checked by our framework. Specifically, it assesses the correctness of the AV's prioritization, trajectory prediction, driving decisions related to NPCs, and speed planning. For instance, to identify an improper 'overtake' decision, we define the specification as: $ImpropOvtkDecn(x) := (av.OnJct \vee av.OnCswk) \wedge DecnOvtk(x)$, which means that if the AV decides to overtake an NPC while near an intersection or on a crosswalk, the 'overtake' decision is considered improper. In this case, x refers to perceived objects, such as vehicles, bicycles, or pedestrians. It is important to note that if a specification is satisfied, a vulnerability has been identified. The detailed specifications can be found on our website [6].

Causal Events Deduction. To identify the causes of accidents from the simplified accident recordings, we design a tool called the Causality Analysis Tool, or CAT for short. CAT analyzes frames labeled as safety-critical to determine whether the planning trajectory could intersect with other traffic participants in a way that might cause an accident. If CAT identifies a potential accident scenario, it analyzes the events leading up to that moment and identifies the actions or behaviors that contributed to the scenario. It is worth noting that even if the AV changes its planning in response to a potential accident scenario, incorrect behavior at that moment could waste valuable reaction time and increase the risk of an accident.

To achieve this, our tool analyzes ST graphs depicting the AV's planning states to discover potential causal events. Based on the Frenet frame method, the ST graph provides a visual way to describe longitudinal behavioral and motion planning. Besides directly presenting whether the trajectory plan is collision-free, the ST graph also describes aspects of the AV's driving decisions and speed planning. Specifically, in an ST graph, time is the horizontal axis, the planned longitudinal trajectory distance is the vertical axis and the planned longitudinal trajectory is a curve. Each point on the curve represents a waypoint on the planned trajectory, and the curve's gradient represents the speed. The motion of other traffic participants can be drawn as rectangles that block certain parts of the AV's longitudinal path during a specific time interval. An ideal speed curve intersects with none of these rectangles so that there is no collision between the AV and NPCs. The positional relationship between the speed curve and an obstacle block in the ST graph presents the AV's behavioral planning result for the related traffic participant. If the obstacle block of a traffic participant is above/below the AV's speed curve, the driving decision by the AV is to yield/overtake, as shown in Figure 4. Therefore, for achieving collision-free trajectory planning, it is imperative that the vehicle accurately perceives all surrounding NPCs and predicts their future trajectories with high precision. This ensures that there is no overlap between the AV and NPCs at each time step. Fundamentally, this planning process equates to solving a constraint satisfaction problem, where the constraints are defined by the drivable area. In an ideal scenario, precise outputs from the perception and prediction modules would enable the computation to guarantee a collision-free trajectory.

Our tool performs a detailed comparison and analysis of the ST graph from the AV perspective against the ground truth, frame by frame. The idea is that for any given frame in the recording, CAT can reconstruct accurate subsequent trajectories of NPCs using data from the future segments of the recording. This reconstructed trajectory is then treated as the ground truth for assessing the effectiveness of the prediction module. Additionally, we examine the planning module of the tool to verify whether it accurately calculates the necessary constraints for ensuring collision-free trajectory planning for the respective frame.

The analysis process of CAT is shown in Figure 5. CAT firstly checks the priority prediction of the NPC involved in the accident. If the NPC's priority prediction is 'ignore', it means that the cause of the collision is wrong priority prediction. This is because AVs do not consider an ignored NPC in the subsequent planning. This omission manifests as a lack of black blocks representing calculated constraints in the ST graph for the NPC, with only the blue blocks indicating the ground truth constraints present. If the AV's speed planning curve does not intersect with the obstacle blocks by the AV but intersects with the obstacle block in the ground truth, it means that the cause of the collision is the AV's misunderstanding of the NPC's future action. This situation is characterized by a significant deviation in the ST graph for the NPC, where there is a clear discrepancy between the constraints calculated by the AV and those of the ground truth.

If the prediction of the NPC made no error, CAT checks the AV's behavioral planning and then the motion planning. In the potential safety-critical frame identification step, ACAV filters potential improper driving decisions made by the AV. When CAT checks the behavioral planning result, if the speed curve intersects with any obstacle blocks near the risky driving decision, it means that improper behavioral planning is to blame for the accident. For example, the speed curve in the ST graph improperly extends beyond an NPC's block to overtake it. However, in this particular scenario, the AV is unable to find a viable trajectory to avoid a collision with another NPC. If the speed curve still intersects with other obstacle blocks based on reasonable behavioral planning, it means that improper motion planning caused by risky speed limits is to blame for the accident. In this case, the speed curve in the ST graph demonstrates an insufficient margin relative to the NPC's block, indicating a lack of adequate space to safely avoid the NPC. If CAT finds that the AV's planning is collision-free, it compares the actual trajectory with the planned trajectory. If there is a deviation between the two trajectories, we can infer that the AV failed to execute the planning due to being out of control (e.g., due to skidding).

Generalizability. While we have presented ACAV in the context of Apollo, the overall approach can be generalized to other ADSs, given that it operates solely on accident recordings and does not require knowledge of the specific internal designs of the systems involved in generating the recordings. The primary assumption for employing our framework is thus the ability to generate/obtain similar recordings. Fortunately, modern ADSs typically have multi-module architectures similar to that of Apollo.

We illustrate the generalizability of ACAV by applying it to recordings obtained from the Autoware.universe ADS [3] and the Carla simulator [23]. We systematically examined the semantic structure of message fields required by ACAV from various modules, including localization, perception, and planning modules. In the case of localization messages, there were similarities between the fields in Autoware.universe and Apollo. Meanwhile, the perception module in Autoware.universe contained tasks related to detecting nearby obstacles and predicting their future trajectories, a functionality akin to the combined roles of perception and prediction modules in Apollo. Nonetheless, some disparities arose in the message structure. Notably, Autoware.universe lacked an obstacle priority field within perception messages and a behavioral planning field within planning messages. To mitigate these differences in message format, we populated the missing fields with default values. As a result, ACAV demonstrated the capability to identify causal events such as wrong trajectory prediction, incorrect speed planning, and instances of vehicles going out of control. However, it was unable to identify causal events related to incorrect priority prediction

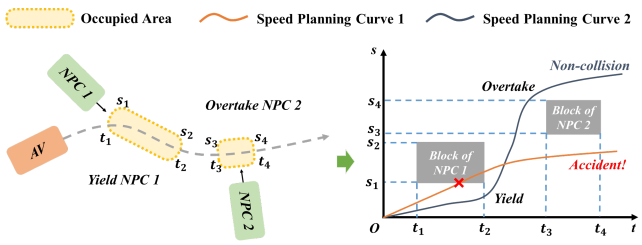


Figure 4: Speed planning based on an ST graph

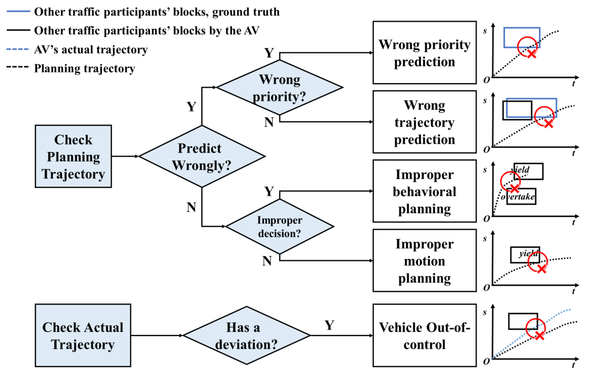

Other traffic participants' blocks, ground truth
Other traffic participants' blocks by the AV
AV's actual trajectory
Planning trajectory
Check Planning Trajectory
Predict Wrongly?
Wrong priority?
Improper decision?
Wrong priority prediction
Wrong trajectory prediction
Improper behavioral planning
Improper motion planning
Check Actual Trajectory
Has a deviation?
Vehicle Out-of-control
yield
overtake

Figure 5: CAT for causality analysis

and erroneous behavioral planning due to the absence of corresponding fields in the message structure (an issue that requires an engineering effort in Autoware.universe to solve).

4 EVALUATION

4.1 Research Questions & Evaluation Metrics

To evaluate the performance of our framework, we conducted experiments to answer the following research questions:

- **RQ1**: Which combination of weights for feature vector categories and which threshold in the "segmenting and pruning" phase are the most effective?
- **RQ2**: Does ACAV effectively simplify accident recording compared to other approaches?
- **RQ3**: How many different causal events can the causality analysis of ACAV automatically identify?
- **RQ4**: To what extent can ACAV accurately identify causal events?

For RQ1 and RQ2, we evaluated the performance of the simplification methods used in the first stage based on two metrics: the 'ratio of reduced frames' and the 'recall of critical frames'. The ratio of reduced frames refers to the length of the removed driving recording over the length of the driving recording before the accident, whereas the recall of critical frames is the number of critical frames in the reduced recording segment over that in the entire recording. Since the subsequent causality analysis relies on these critical frames, we aimed to preserve them as much as possible. Therefore, we initially focused on the recall metric of different methods and then considered their ratio of reduced frames. For RQ3, we assessed the effectiveness of the ACAV by analyzing the number of different causal events it could automatically identify based on the simplification of accident recordings. For RQ4, we evaluated the accuracy of our framework in identifying causal events resulting from versions of Apollo injected with specific faults.

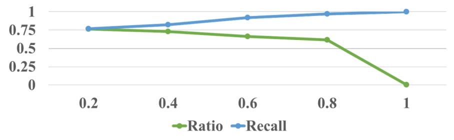


Figure 6: Ratio *(higher is better)* and recall *(higher is better)* of the pruning method under different thresholds

Table 5: Ratio *(higher is better)* and recall *(higher is better)* of different combinations of voting methods

Weight Ratio (map:perc:pln)	1:1:0	1:0:1	0:1:1	**1:1:1**	2:1:1	1:2:1	**1:1:2**
Ratio	74.64%	96.43%	74.64%	**50.03%**	60.26%	74.64%	**62.23%**
Recall	79.62%	11:06%	79.62%	**93.01%**	89.19%	79.62%	**94.41%**

4.2 Experiments and Discussion

4.2.1 RQ1. Different segmenting methods lead to different segmentations of the recording, which can affect the efficacy of test reduction and the final analysis. This is due to the varying contributions of features in depicting a driving scenario. Additionally, using the same contribution for all features can result in many short clips and a lower reduction ratio of original recordings. To design a coarse-grained test reduction method, we evaluated the effectiveness of various combinations of weights assigned to categories of feature vectors and the threshold for determining accident recording segments in RQ1. This method aims to identify and remove non-accident segments to reduce the overall size of the recording for analysis.

We first evaluated the performance of different settings of the voting method for frame segmenting when the threshold value was set as 0.8 for pruning and present the results in Table 5. We focused on the recall of critical frames, as this factor can significantly impact the causality analysis conducted by our a priori frame checker and CAT. The results indicated that the voting method with a weight ratio of $map : perc : pln = 1 : 1 : 2$ (i.e., the method adopted by our framework) achieved the best total recall rate of 94.41% across all frame segmenting methods. This method also had a reduced frame ratio of 62.23%, signifying its effectiveness in removing non-accident recording segments from the analysis. It is also worth noting that the voting method with a weight ratio of $map : perc : pln = 1 : 1 : 1$ achieved a similar recall rate (93.01%) compared to our method (94.41%) while having the lowest ratio of reduced frames among all the weight combinations. However, we observed that the segments generated by this weight combination were fewer in number and larger in length than those created by our segmenting method, leading to fewer segments being discarded in the recording pruning stage. As a result, insufficient recording pruning allowed this method to maintain a promising recall, but it does not necessarily imply that this is an effective segmenting

Table 6: Ratio *(higher is better)* and recall *(higher is better)* of different recording segmenting methods

	ACAV	STRaP	Length: 4s	Length: 8s	Length: 12s	Length: 16s
Ratio	62.23%	60.57%	76.74%	54.40%	32.64%	16.29%
Recall	94.41%	30.81%	72.26%	82.92%	86.85%	91.35%

method. The optimal balance between recall and pruning efficiency is crucial for an effective segmenting method, and our method with the weight ratio $map : perc : pln = 1 : 1 : 2$ has demonstrated better overall performance in capturing critical frames and pruning irrelevant ones.

In order to determine the optimal threshold for our segment pruning method, we conducted a series of experiments, adjusting the threshold for identifying accident-related segments in increments of 0.2, starting from 0.2. We focused on the same two metrics: recall and ratio. The results presented in Figure 6 reveal that as the threshold value increases, recall progressively improves. When the threshold value exceeds 0.4, recall consistently remains above 80%. Simultaneously, the ratio gradually decreases as the threshold value rises. From a threshold of 0.2 to 0.8, the ratio experiences minimal change and maintains a level above 60%. However, when the threshold increases from 0.8 to 1, the ratio experiences a substantial decrease compared to previous levels. Based on these findings, we concluded that a threshold of 0.8 is optimal, as it strikes a balance between high record reduction performance and the retention of a sufficient number of safety-critical frames.

4.2.2 RQ2. For RQ2, our objective is to compare our accident recording simplification method with a variety of alternative fixed-length recording pruning methods and the STRaP framework [21], an AV recording simplification method. We set the lengths at 4, 8, 12, and 16 seconds before the accident, considering that the remaining segment length of our approach is approximately between 4s and 16s. The results of our experiment are displayed in Table 6. The rows represent the evaluation metrics of different segmenting and pruning methods, while the columns indicate the various accident categories included in the experiments. A comparison with fixed-length segmenting methods reveals that it is not feasible to establish a fixed remaining length that effectively balances a substantial reduction ratio with a high critical frame recall. Upon further examination of the accident-related segment lengths, we believe that the primary reason for this outcome is the variability in the duration of interaction between the AV and the NPC involved in different accidents. This observation also highlights the utility and generalizability of our approach, which can adapt to a wide range of cases.

As our segmenting and pruning method shares similar goals with the concept of test reduction and prioritization, we further compared our accident recording simplification method with STRaP, which scales redundant segments with similar contents down to a given length to reduce the length of a recording. As shown in RQ1, ACAV's ratio of reduced frames is 62.23% on average. Therefore, we restricted the retained recording length in STRaP as 40% of the number of frames in the original segment to ensure a similar reduced frame ratio, i.e., a ratio rate of about 60%. In our experiment, STRaP achieved a total reduced frame ratio rate of 39.43% and a recall rate of 30.81%. The reason is that the STRaP framework, while effective in its intended purpose, modifies the content of the original recordings in such a way that distorts the temporal relationships between events and their true durations. This alteration of the original recordings makes STRaP unsuitable for causality analysis.

Table 7: The number of causal events over different accident types

	Wrong Priority Prediction	Wrong Trajectory Prediction	Wrong Behavioral Planning	Wrong Motion Planning	Vehicle Out-of-control
Total	**26**	**51**	**17**	**67**	**103**
Intersection	0	0	6	27	39
Merging	20	27	4	23	30
Tailgating	6	24	7	17	34

4.2.3 RQ3. In RQ3, our objective is to determine ACAV's performance on the accident recordings collected for the original ADS. To achieve this, we conducted a comprehensive evaluation by applying our framework to a dataset comprising 110 accident recordings, all generated by an AV testing engine [63]. This dataset encompassed a variety of accident scenarios, including 43 intersection accidents, 31 merging accidents, and 36 rear-end accidents. ACAV successfully identified the causal events for 103 of these accident recordings. However, our study found that ACAV was unable to detect any significant causal events in 7 accident recordings. Upon further examination, we discovered that these accidents merely involved minor scratches between the AV and the NPC, without any severe impacts taking place. This issue can be attributed to the limitations of computational precision, which can be perceived as an engineering challenge arising from the complexities of accurately processing distances.

For the remaining 103 accidents, we conducted a manual verification process. This entailed revisiting all the causality analysis reports by replaying the accident recordings and validating the causal events identified by ACAV. In particular, we conducted a systematic examination of all causal events identified by our framework and present the specific numbers for each accident type in Table 7. These results indicate that ACAV can effectively identify multiple causal events in various accidents, utilizing each causal event defined by CAT. Through ACAV, the events of wrong trajectory prediction were primarily found in merging and rear-end type accidents, while wrong speed planning events occurred more frequently in intersection and merging accidents. It is important to note that all the accidents in our dataset occurred in rainy or snowy weather conditions, which explains the "vehicle out-of-control" event appearing in all 103 accidents.

4.2.4 RQ4. In RQ4, we sought to assess the accuracy of our framework in identifying the causes of accidents. To achieve this, we injected eight distinct fault types, as detailed in Table 8, into the ADS. Specifically, F1 can cause an accident due to wrong priority prediction causal events, while F2 causes accidents based on wrong trajectory prediction. Conversely, F3 through F7 are designed to cause accidents due to improper behavioral planning. Finally, F8 is identified as the trigger for a causal event related to improper motion planning. For each fault type, we ran the testing engine [63]

Table 8: The eight types of faults injected into the customized ADS

Fault Type	Location	Description
F1	`AssignIgnoreLevel()@obstacle_prioritizer.cc`	Assign 'ignore' priority to all the detected NPCs by default.
F2	`PredictObstacle()@predictor_manager.cc`	Assign improper trajectory prediction models to NPCs to get erroneous trajectory prediction.
F3	`MakeStaticObstacleDecision()@path_decider.cc`	Make 'ignore' decisions to all the static NPCs near the AV's planned trajectory.
F4	`MakeObjectDecision()@speed_decider.cc`	Make 'follow' decisions to any NPCs in front of the AV which tend to stop, instead of 'stop' decisions or changing lanes.
F5	`MakeObjectDecision()@speed_decider.cc`	Make 'ignore' decisions to an NPC ahead of the AV, if the AV is not following or keeping distance from it.
F6	`MakeObjectDecision()@speed_decider.cc`	Make 'yield' decisions to a high-speed NPC accelerating ahead of the AV, which leads to AV's low speed in a fast lane.
F7	`MakeObjectDecision()@speed_decider.cc`	Make 'overtake' decisions to any NPC if it is near the AV.
F8	`GetSpeedLimits()@speed_limit_decider.cc`	Keep a high speed even being close to NPCs.

Table 9: Precision *(higher is better)*, Recall *(higher is better)*, and Accuracy *(higher is better)* of causal events over accidents with different fault injections

Location	Prediction Module			Planning Module						
Fault Types	F1	F2	Total	F3	F4	F5	F6	F7	F8	Total
Numbers	155	126	281	132	146	202	166	145	134	925
Precision (%)	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Recall (%)	100.00	90.00	95.97	87.61	73.98	89.70	86.72	77.19	77.19	82.56
Accuracy (%)	100.00	92.06	**96.44**	89.39	**78.08**	91.58	89.76	82.07	80.60	**85.73**

for approximately one day and recorded the resulting accidents. It is imperative to highlight our efforts to ensure the complexity of each recorded test case. We accomplished this by implementing varying extended routes and incorporating multiple NPCs of diverse types. Furthermore, we standardized the duration of each recording file to 120 seconds. In total, we amassed a dataset comprising 1206 accident recordings.

Subsequently, we applied our framework to analyze these accident recordings, documenting the accident causal events and their respective time frames. In this experiment, if a causal event's duration significantly surpassed those of other causal events, we deemed it to be the 'main' cause of the given accident. For example, in the case of fault F2, if the injected fault takes effect, it should persist for a sufficient duration to accumulate a noticeable trajectory prediction error, which is crucial for causing accidents. Consequently, the associated causal event, namely, 'wrong trajectory prediction', would be identified in the recording files as the main cause of this fault. If our framework correctly identifies the functions in line with the injected faults, we conclude that our framework accurately determines the cause of the accidents.

As shown in Table 9, ACAV performs well, accurately identifying causal events in 1064 out of the accident recordings, with a precision of 100.00% for both the prediction and planning modules. This indicates that, for a specific type of fault, our framework can both precisely identify the causal events within the recording and distinguish recordings that do not include these causal events. Furthermore, this is complemented by a recall rate of 95.97% and an accuracy of 96.44% in the prediction module, along with a recall rate of 82.56% and an accuracy of 85.73% in the planning module.

Nevertheless, our investigation uncovered a limitation, as ACAV failed to detect the causal events in 142 accident recordings. Upon a more in-depth examination, we discovered that when faults are injected into the planning module, two or three closely interrelated causal events often occur simultaneously. For instance, in 15 accidents linked to fault F7, an additional causal event surfaced: the vehicle going out of control. This event was attributable to the elevated speed requirement of the 'overtake' decision, particularly evident during inclement weather conditions like rain or snow. We observed that ACAV successfully identified interrelated causal events in 105 out of the 142 accidents.

4.2.5 Threats to Validity. We acknowledge certain limitations and threats to the validity of our evaluation. While our approach has been implemented for two distinct platforms—Apollo, simulated with the SVL Simulator, and Autoware.universe, simulated with Carla—our evaluation is exclusively focused on the Apollo ADS. The reason is that there is currently no suitable fuzzing engine implemented for Autoware.universe. This absence presents a challenge in acquiring sufficient accident recordings for a comprehensive evaluation of our approach on the platform. Second, during testing, we observed that the AV primarily considered NPCs in front of it when planning driving behavior. When an NPC hits the AV from behind, ACAV may not yield effective analysis results. This issue could be addressed by incorporating more intelligent NPC behavior configurations in the simulator, which would better emulate interactions between real-world vehicles. Furthermore, it is generally accepted that the rear vehicle should bear more responsibility in a rear-end accident, a principle that is also practiced in many jurisdictions [5]. Third, it is imperative to acknowledge that the faults injected in RQ4 do not reflect the real-world faults in the ADSs. However, the resulting accidents from these injected faults are similar to those caused by real-world faults in ADSs, lending credence to our framework's ability to accurately identify causal factors of accidents. Moreover, the inherent complexity of ADSs, attributed to their reliance on logic-based code, external dependency libraries, and machine learning-based models across various modules, contributes to a significant challenge for repairing. As reported in a study [30], more than half of the AV faults originate from incorrect algorithmic implementations or configurations, often involving extensive code segments exceeding 20 lines. Consequently, while our framework can interpret accident recordings and pinpoint potential causes, it should not be considered a panacea for repairing the underlying bugs in ADS systems.

5 RELATED WORK

System-level testing for AVs is designed to evaluate the performance of the entire ADS, as opposed to module-level testing, which focuses on individual modules or specific functionalities. This comprehensive evaluation is achieved through the use of scenario-based test cases and test oracles. Current research in system-level testing primarily focuses on generating corner cases and error-prone driving scenarios. There are two main categories of scenario sources: real-world data, and testing frameworks.

One category of work generates scenarios derived from scenarios observed in the real world, emphasizing the similarity between the generated scenarios and real-world ones [43, 44]. Zhang et al. [58] proposed a method based on 3D scene reconstruction, which uses images collected by the in-vehicle camera to recreate scenarios as test cases. Gambi et al. [25] proposed AC3R, which extracts information from collision reports and constructs new test scenarios using simulation methods. DEEPCRASHTEST [12] recreates accident scenarios based on accident videos. Fremont et al. [29] combined formal verification with clustering algorithms to select usable test scenarios. There is also an approach [48] that evaluates the performance of the ADSs by comparing them with that of human drivers according to features extracted from real-world scenarios.

Another category of work generates scenarios by using a (domain-specific) testing framework. Two widely-adopted methodologies are search-based or sampling-based methods [7, 8, 13, 24, 32, 40, 47, 64]. Search-based methods, or fuzzing, typically search the parameter space for specific parameter values to achieve a certain testing goal. To guarantee the efficiency of the heuristic search method adopted, e.g., genetic algorithms, a well-defined fitness function is required. Althoff et al. [9] defined a calculating metric, the drivable area, to quantify the search of solution space, and combined reachability analysis with optimization techniques to obtain test scenarios. Li et al. [38] proposed AVFuzzer, which uses safety potential, the distance between the ego vehicle and other traffic participants, as the fitness function for a genetic algorithm-based fuzzer to find scenarios that could lead to collisions. Combining program analysis techniques and evolutionary algorithm-based fuzzing, PlanFuzz [54] defines behavioral planning vulnerability distance as the guidance for the generation of test scenarios that would cause the autonomous vehicle to stop under safe conditions. Sun et al. [52] defined a metric for quantifying the degree to which autonomous vehicles violate traffic rules in a driving scenario, guiding their fuzzer to generate test cases that violate traffic regulations. Sampling-based methods sample from a naturalistic scenario distribution to generate test cases. A series of works [33, 56, 61, 62] has studied sampling in different driving scenarios based on importance sampling [53]. Batsch et al. [15] built a Gaussian process classification model to estimate the safety of a scenario probabilistically, with the training data sampled from simulation-generated traffic congestion scenarios. NADE [28] collects driving scenarios from real-world data and samples to generate realistic and safety-critical scenarios.

The aforementioned works primarily concentrate on evaluating the performance of ADSs comprehensively and identifying new vulnerabilities. However, their focus lies in determining whether the ADSs fail to meet the test oracles, rather than understanding the underlying reasons for these failures. Our method is driven by the goal of analyzing the actual cause of safety violations, such as collisions, by concentrating on the testing process itself.

In recent years, causality has become a widely-adopted methodology to analyze complex systems. Forney et al. [35] proposed an interactive platform for fault diagnosis and forensic investigation in fields such as airplane accidents. Bareinboim et al. [10] proposed a causal inference-based method to solve data fusion problems in the context of big data. Biebl et al. [14] presented a causal model to predict accident risks in an intersection for drivers with impairments. In addition to works focusing on AI [11, 18, 19, 59], some works have applied causality to the security analysis of CPSs [34, 36, 37, 41, 42]. Zhang et al. [60] monitored, inspected, and located anomalies in industrial control systems using a causal model based on maximum information coefficient and transfer entropy. Poskitt et al. [46] proposed a causality-guided fuzzing method that identifies and generalizes the causality of events in testing to find new test cases with different causal relationships. Our method is designed to employ causality analysis on autonomous driving accident records to facilitate deeper fault analysis and uncover the underlying causes of accidents. By examining the causal factors that contribute to accidents, we can better understand the limitations and vulnerabilities of autonomous driving systems. This, in turn, allows engineers to make more targeted improvements, enhance safety, and reduce the likelihood of similar accidents occurring in the future.

6 CONCLUSION

We presented ACAV, an automated framework for determining the causal events in AV accidents. We successfully implemented it in both Apollo and Autoware.universe and evaluated our framework using 110 accident driving recordings from the Baidu Apollo ADS, successfully identifying causal events in 103 of them. After analyzing 1206 accident recordings collected from ADSs injected with specific faults, we further showed that it identifies causal events correctly.

In future work, we are interested in developing automatic program repair techniques for ADSs, leveraging the results of causality analyses from accidents. By incorporating these advancements, we hope to create a comprehensive framework that can contribute significantly to the safety and reliability of AVs in real-world scenarios.

ACKNOWLEDGMENT

We extend our sincere appreciation to the three anonymous referees from ICSE for their invaluable insights and constructive feedback. This research was jointly sponsored by Shanghai Sailing Program under Grants No. 23YF1427500, the NSFC Program under Grants No. 62302304, ShanghaiTech Startup Funding, and the Ministry of Education, Singapore under its Academic Research Fund Tier 3 (Award ID: MOET32020-0004). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore.

REFERENCES

[1] 2019. Fatal Tesla Crash Exposes Gap In Automaker's Use Of Car Data. https://www.forbes.com/sites/alanohnsman/2018/04/16/tesla-autopilot-fatal-crash-waze-hazard-alerts/?sh=229d9e685572. Online; accessed December 2023.
[2] 2021. Apollo 7.0. https://github.com/ApolloAuto/apollo/releases/tag/v7.0.0. Online; accessed April 2023.
[3] 2021. Autoware.universe galactic. https://github.com/autowarefoundation/autoware.universe/tree/galactic. Online; accessed November 2023.
[4] 2022. Fatal crash on SB I-680 onramp in San Jose. https://www.kron4.com/news/bay-area/fatal-crash-on-sb-i-680-onramp-in-san-jose/. Online; accessed December 2023.
[5] 2022. Rear-End Collisions: Fault & Compensation. https://www.forbes.com/advisor/legal/auto-accident/rear-end-collision/#establishing_fault_for_rear_end_accidents_section. Online; accessed December 2023.
[6] 2023. ACAV2023. https://acav2023.github.io. Online; accessed December 2023.
[7] Raja Ben Abdessalem, Shiva Nejati, Lionel C Briand, and Thomas Stifter. 2018. Testing vision-based control systems using learnable evolutionary algorithms. In

Proceedings of the 40th International Conference on Software Engineering. 1016–1026.

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

[9] Matthias Althoff, Markus Koschi, and Stefanie Manzinger. 2017. CommonRoad: Composable benchmarks for motion planning on roads. In *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 719–726.

[10] Elias Bareinboim and Judea Pearl. 2016. Causal inference and the data-fusion problem. *Proceedings of the National Academy of Sciences* 113, 27 (2016), 7345–7352.

[11] Elias Bareinboim and Jin Tian. 2015. Recovering causal effects from selection bias. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 29.

[12] Sai Krishna Bashetty, Heni Ben Amor, and Georgios Fainekos. 2020. Deepcrashtest: Turning dashcam videos into virtual crash tests for automated driving systems. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 11353–11360.

[13] Halil Beglerovic, Michael Stolz, and Martin Horn. 2017. Testing of autonomous vehicles using surrogate models and stochastic optimization. In *ITSC*. IEEE, 1–6.

[14] Bianca Biebl, Severin Kacianka, Anirudh Unni, Alexander Trende, Jochem W. Rieger, Andreas Lüdtke, Alexander Pretschner, and Klaus Bengler. 2021. A Causal Model of Intersection-Related Collisions for Drivers With and Without Visual Field Loss. In *HCI International 2021 - Late Breaking Papers: HCI Applications in Health, Transport, and Industry - 23rd HCI International Conference, HCII 2021, Virtual Event, July 24-29, 2021 Proceedings (Lecture Notes in Computer Science, Vol. 13097)*, Constantine Stephanidis, Vincent G. Duffy, Heidi Krömker, Fiona Fui-Hoon Nah, Keng Siau, Gavriel Salvendy, and June Wei (Eds.). Springer, 219–234. https://doi.org/10.1007/978-3-030-90966-6_16

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

[16] California Department of Motor Vehicles. 2022. CA Driver's Handbook. https://www.dmv.ca.gov/portal/handbook/california-driver-handbook/. Online; accessed December 2022.

[17] California Legislative Counsel Bureau. 2022. Rules of the Road. https://leginfo.legislature.ca.gov/faces/codes_displayexpandedbranch.xhtml?tocCode=VEH&division=11.&title=&part=&chapter=&article=&nodetreepath=15. Online; accessed April 2023.

[18] Juan Correa and Elias Bareinboim. 2017. Causal effect identification by adjustment under confounding and selection biases. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.

[19] Juan Correa, Jin Tian, and Elias Bareinboim. 2018. Generalized adjustment under confounding and selection biases. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[20] Subasish Das, Anandi Dutta, and Ioannis Tsapakis. 2020. Automated vehicle collisions in California: Applying Bayesian latent class model. *IATSS research* 44, 4 (2020), 300–308.

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

[22] Vinayak V Dixit, Sai Chand, and Divya J Nair. 2016. Autonomous vehicles: disengagements, accidents and reaction times. *PLOS ONE* 11, 12 (2016), 1–14.

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

[24] Tommaso Dreossi, Alexandre Donzé, and Sanjit A Seshia. 2019. Compositional falsification of cyber-physical systems with machine learning components. *Journal of Automated Reasoning* 63 (2019), 1031–1053.

[25] Tommaso Dreossi, Daniel J Fremont, Shromona Ghosh, Edward Kim, Hadi Ravanbakhsh, Marcell Vazquez-Chanlatte, and Sanjit A Seshia. 2019. VerifAI: A toolkit for the formal design and analysis of artificial intelligence-based systems. In *Computer Aided Verification: 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I 31*. Springer, 432–442.

[26] Haoyang Fan, Fan Zhu, Changchun Liu, Liangliang Zhang, Li Zhuang, Dong Li, Weicheng Zhu, Jiangtao Hu, Hongye Li, and Qi Kong. 2018. Baidu Apollo EM Motion Planner. *arXiv preprint arXiv:1807.08048* (2018).

[27] Francesca M Favarò, Nazanin Nader, Sky O Eurich, Michelle Tripp, and Naresh Varadaraju. 2017. Examining accident reports involving autonomous vehicles in California. *PLoS one* 12, 9 (2017), e0184952.

[28] Shuo Feng, Xintao Yan, Haowei Sun, Yiheng Feng, and Henry X Liu. 2021. Intelligent driving intelligence test for autonomous vehicles with naturalistic and adversarial environment. *Nature communications* 12, 1 (2021), 748.

[29] Daniel J Fremont, Edward Kim, Yash Vardhan Pant, Sanjit A Seshia, Atul Acharya, Xantha Bruso, Paul Wells, Steve Lemke, Qiang Lu, and Shalin Mehta. 2020. Formal scenario-based testing of autonomous vehicles: From simulation to the real world. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 1–8.

[30] Joshua Garcia, Yang Feng, Junjie Shen, Sumaya Almanee, Yuan Xia, Chen, and Qi Alfred. 2020. A Comprehensive Study of Autonomous Vehicle Bugs. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering* (Seoul, South Korea) *(ICSE '20)*. Association for Computing Machinery, New York, NY, USA, 385–396. https://doi.org/10.1145/3377811.3380397

[31] David González, Joshué Pérez, Vicente Milanés, and Fawzi Nashashibi. 2015. A review of motion planning techniques for automated vehicles. *IEEE Transactions on intelligent transportation systems* 17, 4 (2015), 1135–1145.

[32] Florian Hauer, Tabea Schmidt, Bernd Holzmüller, and Alexander Pretschner. 2019. Did We Test All Scenarios for Automated and Autonomous Driving Systems?. In *ITSC*. IEEE, 2950–2955.

[33] Zhiyuan Huang, Henry Lam, and Ding Zhao. 2017. An accelerated testing approach for automated vehicles with background traffic described by joint distributions. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 933–938.

[34] Amjad Ibrahim, Severin Kacianka, Alexander Pretschner, Charles Hartsell, and Gabor Karsai. 2019. Practical Causal Models for Cyber-Physical Systems. In *NASA Formal Methods - 11th International Symposium, NFM 2019, Houston, TX, USA, May 7-9, 2019, Proceedings (Lecture Notes in Computer Science, Vol. 11460)*, Julia M. Badger and Kristin Yvonne Rozier (Eds.). Springer, 211–227. https://doi.org/10.1007/978-3-030-20652-9_14

[35] Amjad Ibrahim, Tobias Klesel, Ehsan Zibaei, Severin Kacianka, and Alexander Pretschner. 2020. Actual causality canvas: a general framework for explanation-based socio-technical constructs. In *ECAI 2020*. IOS Press, 2978–2985.

[36] Zahra Jadidi, Joshua Hagemann, and Daniel Quevedo. 2022. Multi-step attack detection in industrial control systems using causal analysis. *Computers in Industry* 142 (2022), 103741. https://doi.org/10.1016/j.compind.2022.103741

[37] Severin Kacianka, Amjad Ibrahim, and Alexander Pretschner. 2020. Expressing Accountability Patterns using Structural Causal Models. *CoRR* abs/2005.03294 (2020). arXiv:2005.03294 https://arxiv.org/abs/2005.03294

[38] Guanpeng Li, Yiran Li, Saurabh Jha, Timothy Tsai, Michael Sullivan, Siva Kumar Sastry Hari, Zbigniew Kalbarczyk, and Ravishankar Iyer. 2020. AV-Fuzzer: Finding safety violations in autonomous driving systems. In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 25–36.

[39] Guannan Lou, Yao Deng, Xi Zheng, Mengshi Zhang, and Tianyi Zhang. 2022. Testing of Autonomous Driving Systems: Where Are We and Where Should We Go?. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Singapore, Singapore) *(ESEC/FSE 2022)*. Association for Computing Machinery, New York, NY, USA, 31–43. https://doi.org/10.1145/3540250.3549111

[40] Yixing Luo, Xiao-Yi Zhang, Paolo Arcaini, Zhi Jin, Haiyan Zhao, Fuyuki Ishikawa, Rongxin Wu, and Tao Xie. 2021. Targeting requirements violations of autonomous driving systems by dynamic evolutionary search. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 279–291.

[41] Fereidoun Moradi, Sara Abbaspour Asadollah, Ali Sedaghatbaf, Aida Čaušević, Marjan Sirjani, and Carolyn Talcott. 2020. An actor-based approach for security analysis of cyber-physical systems. In *Formal Methods for Industrial Critical Systems: 25th International Conference, FMICS 2020, Vienna, Austria, September 2–3, 2020, Proceedings 25*. Springer, 130–147.

[42] Vivek Nigam, Minyoung Kim, Ian Mason, and Carolyn Talcott. 2022. Detection and diagnosis of deviations in distributed systems of autonomous agents. *Mathematical Structures in Computer Science* 32, 9 (2022), 1254–1282. https://doi.org/10.1017/S0960129522000251

[43] Philippe Nitsche, Pete Thomas, Rainer Stuetz, and Ruth Welsh. 2017. Pre-crash scenarios at road junctions: A clustering method for car crash data. *Accid. Anal. Prev.* 107 (2017), 137–151.

[44] Jan-Pieter Paardekooper, S Montfort, Jeroen Manders, Jorrit Goos, E de Gelder, O Camp, O Bracquemond, and Gildas Thiolon. 2019. Automatic identification of critical scenarios in a public dataset of 6000 km of public-road driving. In *ESV*.

[45] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. 2016. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles* 1, 1 (2016), 33–55.

[46] Christopher M. Poskitt, Yuqi Chen, Jun Sun, and Yu Jiang. 2023. Finding Causally Different Tests for an Industrial Control System. In *Proc. IEEE/ACM International Conference on Software Engineering (ICSE'23)*. IEEE, 2578–2590.

[47] Vincenzo Riccio and Paolo Tonella. 2020. Model-based exploration of the frontier of behaviours for deep learning system testing. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 876–888.

[48] Christian Roesener, Felix Fahrenkrog, Axel Uhlig, and Lutz Eckstein. 2016. A scenario-based assessment approach for automated driving by using time series classification of human-driving behaviour. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 1360–1365.

[49] Guodong Rong, Byung Hyun Shin, Hadi Tabatabaee, Qiang Lu, Steve Lemke, Mārtiņš Možeiko, Eric Boise, Geehoon Uhm, Mark Gerow, Shalin Mehta, et al. 2020. LGSVL Simulator: A high fidelity simulator for autonomous driving. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 1–6.

[50] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. 2018. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems* 1 (2018), 187–210.

[51] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. 2017. On a Formal Model of Safe and Scalable Self-driving Cars. *CoRR* abs/1708.06374 (2017). arXiv:1708.06374 http://arxiv.org/abs/1708.06374

[52] Yang Sun, Christopher M Poskitt, Jun Sun, Yuqi Chen, and Zijiang Yang. 2022. LawBreaker: An Approach for Specifying Traffic Laws and Fuzzing Autonomous Vehicles. In *37th IEEE/ACM International Conference on Automated Software Engineering*. 1–12.

[53] Surya T Tokdar and Robert E Kass. 2010. Importance sampling: a review. *Wiley Interdisciplinary Reviews: Computational Statistics* 2, 1 (2010), 54–60.

[54] Ziwen Wan, Junjie Shen, Jalen Chuang, Xin Xia, Joshua Garcia, Jiaqi Ma, and Qi Alfred Chen. 2022. Too Afraid to Drive: Systematic Discovery of Semantic DoS Vulnerability in Autonomous Driving Planning under Physical-World Attacks. In *Network and Distributed System Security (NDSS) Symposium, 2022*.

[55] Song Wang and Zhixia Li. 2019. Exploring the mechanism of crashes with automated vehicles using statistical modeling approaches. *PloS one* 14, 3 (2019), e0214550.

[56] Xinpeng Wang, Huei Peng, and Ding Zhao. 2021. Combining reachability analysis and importance sampling for accelerated evaluation of highway automated vehicles at pedestrian crossing. *ASME Letters in Dynamic Systems and Control* 1, 1 (2021).

[57] Moritz Werling, Julius Ziegler, Sören Kammel, and Sebastian Thrun. 2010. Optimal trajectory generation for dynamic street scenarios in a frenet frame. In *2010 IEEE International Conference on Robotics and Automation*. IEEE, 987–993.

[58] Chi Zhang, Yuehu Liu, Danchen Zhao, and Yuanqi Su. 2014. RoadView: A traffic scene simulator for autonomous vehicle simulation testing. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 1160–1165.

[59] Junzhe Zhang and Elias Bareinboim. 2017. Transfer learning in multi-armed bandit: a causal approach. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. 1778–1780.

[60] Renbin Zhang, Zongze Cao, and Kewei Wu. 2020. Tracing and detection of ICS anomalies based on causality mutations. In *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*. IEEE, 511–517.

[61] Ding Zhao, Xianan Huang, Huei Peng, Henry Lam, and David J LeBlanc. 2017. Accelerated evaluation of automated vehicles in car-following maneuvers. *IEEE Transactions on Intelligent Transportation Systems* 19, 3 (2017), 733–744.

[62] Ding Zhao, Henry Lam, Huei Peng, Shan Bao, David J LeBlanc, Kazutoshi Nobukawa, and Christopher S Pan. 2016. Accelerated evaluation of automated vehicles safety in lane-change scenarios based on importance sampling techniques. *IEEE Transactions on Intelligent Transportation Systems* 18, 3 (2016), 595–607.

[63] Yuan Zhou, Yang Sun, Yun Tang, Yuqi Chen, Jun Sun, Christopher M Poskitt, Yang Liu, and Zijiang Yang. 2023. Specification-based Autonomous Driving System Testing. *IEEE Transactions on Software Engineering* 49, 6 (2023), 3391–3410.

[64] Tahereh Zohdinasab, Vincenzo Riccio, Alessio Gambi, and Paolo Tonella. 2021. Deephyperion: exploring the feature space of deep learning-based systems through illumination search. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 79–90.