Synergistic Redundancy: Towards Verifiable Safety for Autonomous Vehicles

Ayoosh Bansal*, Simon Yu*, Hunmin Kim†, Bo Li*, Naira Hovakimyan*, Marco Caccamo‡, Lui Sha*
*University of Illinois Urbana-Champaign {ayooshb2, jundayu2, lbo, nhovakim, lrs}@illinois.edu,
†Mercer University kim_h@mercer.edu, ‡Technical University of Munich mcaccamo@tum.de

***Abstract*—As Autonomous Vehicle (AV) development has progressed, concerns regarding the safety of passengers and agents in their environment have risen. Each real-world traffic collision involving autonomously controlled vehicles has compounded this concern. Open source autonomous driving implementations show a software architecture with complex interdependent tasks, heavily reliant on machine learning and Deep Neural Networks (DNN), which are vulnerable to non-deterministic faults and corner cases. These complex subsystems work together to fulfill the mission of the AV while also maintaining safety. Although significant improvements are being made towards increasing the empirical reliability and confidence in these systems, the inherent limitations of DNN verification create an, as yet, insurmountable challenge in providing deterministic safety guarantees in AV.**

We propose Synergistic Redundancy (SR), a safety architecture for complex cyber-physical systems, *e.g.,* AV. SR provides verifiable safety guarantees against specific faults by decoupling the mission and safety tasks of the system. Simultaneous to independently fulfilling their primary roles, the partially functionally redundant mission and safety tasks are able to aid each other, synergistically improving the combined system. The synergistic safety layer uses only verifiable and logically analyzable software to fulfill its safety tasks, ensuring reliable operation and certifiable behavior. Close coordination with the mission layer allows easier and early detection of safety-critical faults in the system. Due to the separation of mission and safety concerns, simplifying the mission layer's optimization goals improves its design. SR provides a safe architecture for the deployment of high-performance, although inherently unverifiable, machine learning software within its mission layer. In this work, we first present the design and features of the SR architecture and then evaluate the efficacy of the solution above using industrial simulators, focusing on the crucial problem of obstacle existence detection faults in AV. Our system achieves predictable safety limits and deterministic safe behavior when within these limits.

***Index Terms*—Autonomous Vehicles, Software Reliability, Software Safety, Robustness, Fault Tolerance, Cyber-Physical Systems**

I. Introduction

The dream of fully Autonomous Vehicles (AV) is close to turning into reality [1]. Deep learning enabled breakthroughs in environment perception and path planning [2], coupled with the development of advanced computing platforms for AV [3], [4], have forged a path forward to realizing completely autonomous self-driving, *i.e.,* Level 5 autonomy [5]. Full driving automation, when safe and secure, can improve and save lives [6], [7]. Despite the rapid advancements and proliferation of AV prototypes, the technological path to collision-free safe AV remains unclear [8]–[11]. There have been many casualties where autonomous driving systems, albeit incapable of full autonomy, are known or suspected to be a contributing factor in the traffic collisions [12]–[15].

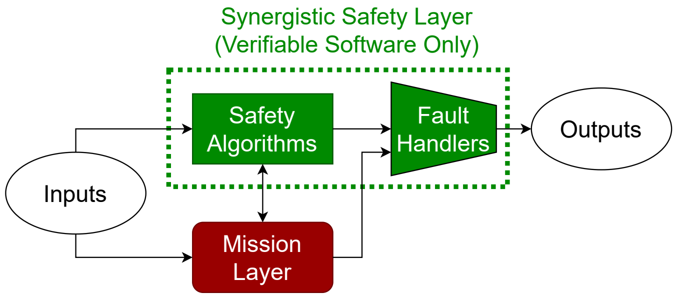


Fig. 1: An overview of SR architecture. The Mission layer is a complex autonomous agent responsible for all tasks required to fulfill the mission of the system, *e.g.,* perception, planning, and control. The Safety layer uses verifiable and logically analyzable software only. It includes Algorithms that provide deterministic guarantees and Fault handlers that determine faults and take corrective actions. The layers exchange intermediate outputs and make available their capabilities to each other to enable cooperative synergies.

Based on available information, these collisions may have been completely avoided, or at the least, their impact reduced had an ideal autonomous driving agent correctly detected the existence of obstacles in the vicinity of the vehicle and responded to them. So how do we get closer to this ideal? We assert that an inherent limitation of traditional autonomous driving agents is that the responsibilities of mission (navigation) and safety (collision avoidance) are diffused throughout the system. Without the separation between mission-critical and safety-critical tasks, the optimization goals of the system become ill-formed. Worse, without separation, the safety problem becomes intermingled and convoluted. Hence in classical control applications, the safety-critical tasks and mission-critical tasks are decoupled in popular architectures [16], [17].

An example of this intermingling are the Deep Neural Networks (DNN) for object detection in AV [2], [18]–[24]. These solutions couple the tasks of identifying object existence and class *e.g.,* vehicle or pedestrian. This approach works incredibly well for mission-critical systems, enabling a plethora of new applications, including AV. However, in safety-critical systems where obstacle existence detection is a necessary requirement, while object classification is not [25], this inter-dependency between the two can result in errors from one task *e.g.,* object classification, to cause critical faults in the other *i.e.,* obstacle existence detection [13]. Furthermore, the intermingled complex tasks then require complex solutions. While DNN has been deployed quite successfully to fulfill these complex requirements, their inherent limitations for complete verification and analysis [10], [11], [26]–[29] make them unsuitable for safety-critical tasks.

This paper presents **Synergistic Redundancy** (SR), a system architecture that decouples mission-critical and safety-critical tasks while simultaneously leveraging close coordination and cooperative synergies between these partially functionally redundant tasks. In SR, a synergistic safety layer[1] implements various logically analyzable and verifiable, therefore certifiable, algorithms as required for safety-critical software [30], [31]. The algorithms fulfill the minimal functionality required to identify, correct or limit safety-critical faults. DNN for perception, planning, localization, and control along with other related tasks, inhabit the mission layer, fulfilling all aspects of the mission of the AV. However, owing to their inherent unverifiability such DNN based solutions can only provide empirical or probabilistic safety guarantees [10], [11], [32]–[34]. Despite this limitation, DNN can be safely integrated within the mission layer of SR due to the safety guarantees of the safety layer. The two layers feature partially overlapping functions with the safety layer fully responsible for providing specific safety guarantees. Figure 1 shows an overview of the SR architecture.

With the differentiated responsibilities, the mission layer's optimization goals become simplified. The mission layer does not need to simultaneously attempt to cover all corner cases for safety and can thus provide high performance for the common case and optimize resource requirements. The safety layer in SR provides specific safety guarantees, independent of faults in the mission layer and the mission layer's availability. Rather than an all-or-nothing approach, the safety layer in SR allows the handling of specific faults, incrementally and monotonically improving the safety of the system. The flexible design helps incremental development and inclusion of solutions to the crucial safety problems in AV.

In traditional systems with mission and safety layer separation [16], [17], the two layers have distinct responsibilities towards the overall system with no direct interactions. In addition to the same separation of responsibilities, SR uniquely leverages close coordination between layers for synergistic improvements for the whole system. As an example, the safety layer bears the primary responsibility for safety and hence overrides mission layer output when a safety-critical fault is detected, however, for each fault it detects, safety-critical or otherwise, the corresponding mission layer module is informed, helping it correct the fault in short term and improve its learned behaviors in the long term. Similarly, owing to the complexity of all the tasks that an AV must undertake, the safety layer can judiciously leverage mission layer capabilities for corrective actions. When safety layer constraints are violated, and no safety-critical faults have been detected, mission layer capabilities can be leveraged to provide a resolution with the least impact on system performance[2].

We apply the SR architecture to address the problem of obstacle existence detection. Safety in AV is a multifaceted issue [1], [35]–[40]. However, real world collisions [12]–[15], [41]–[52] show a recurring pattern of obstacle existence detection faults [25]. Therefore, in this work, we apply SR architecture to address this category of faults while discussing a path to future resolution of other fault types.

We implement and evaluate this example design, using Baidu's Apollo [53] as the mission layer. Within the safety layer, Depth Clustering by Bogoslavskyi and Stachniss [54], [55], is used as the verifiable obstacle detection algorithm. Verification of this algorithm's capabilities, guarantees, and constraints have been established in a prior work [25]. We integrate the obstacle detection algorithm and implement fault handlers within the software-in-the-loop simulation [56] using the LGSVL simulator [57]. LGSVL simulation provides sensing and actuation interfaces to the simulated autonomous agent, with the capability to run custom scenarios. We show that the deterministic guarantees of verifiable algorithms for obstacle detection in the safety layer can be used to protect against respective faults in complex object detectors in the mission layer (§V).

The key contributions of this work are:

- Synergistic Redundancy, an incremental, decoupled, and synergistic safety architecture for complex cyber-physical systems (§III).
- An AV-centric reference design of the SR architecture for the safety-critical function of the obstacle existence detection and frontal collision avoidance (§IV).
- Evaluation of the reference design using software-in-the-loop simulation [56], emulating the obstacle existence detection fault scenario common to real world collisions [25] (§V).[3]

II. Related Work

Redundancy is a crucial concept for creating reliable safety-critical cyber-physical systems (CPS). Full system replication enables a system to tolerate and correct random or systemic faults that an individual system cannot [58]–[65]. Various optimizations for internalizing redundancy to computational hardware have been explored, like lockstep execution on different cores [66], [67], instruction level duplication for CPU [68] or GPU [69] and checkpoint-based enhancements [70]. These techniques reduce the computational overheads of redundant execution, however, the overhead still scales with the size of the complete system, making the overheads expensive, sometimes prohibitively, for large and

[1]For the sake of brevity, going forward the synergistic safety layer is referred to as the safety layer. It should be noted that the synergistic safety layer is different from other safety layer designs as described in Section II.

[2]Table I summarizes the synergistic cooperation in the reference design.

[3]https://github.com/CPS-IL/synergistic_redundancy

complex systems. However, simple redundancy does not address faults shared by all replicas *i.e.* algorithmic or implementation faults [71]–[73]. There are many avenues of such faults occurring in the system and causing disruptions [74], [75]. As powerful a tool as redundancy is, as long as all copies of the system are using the same algorithms or implementations, they will suffer the same algorithmic faults leading to a failure of the CPS. This led to the development of functional or analytical redundancy for fault-tolerant control [76]. Functional redundancy is the use of different systems with different implementations, inputs, capabilities, or algorithms to fulfill the same function *e.g.,* GPS and inertial navigation.

Control Simplex Architecture [16], [17], [77]–[79] applies the concepts of functional redundancy and runtime monitoring to the whole control system. Simplex leverages an independently functional safety controller and a simple decision module. The decision module monitors the system state, switching from a high-performance controller to the safety controller if the system is approaching an unsafe state. Simplex architecture's high performance and safety controllers are fully functional and can independently fulfill the tasks of the system. Simplex architecture is used for many safety-critical control tasks like stability control in avionics. However, creating a safety controller that can fulfill all functions required for an AV's operation without essentially replicating the existing AV software stack has, as yet, not been possible.

Simplex adaptations have been designed for safe control in presence of software and physical faults in AV [80]–[82]. However, the critical problem of perception is not addressed and mission and safety modules remain completely separated. As we show, closer coordination can enable synergistic improvements. The various Simplex controllers can be adopted into the SR architecture for addressing specific faults or providing additional verifiable capabilities. SL1-Simplex [81] would be especially suitable to maintain the safe operating velocity of the AV in presence of dynamic environmental and physical conditions, providing low-level control support for the limits and overrides imposed by the SR safety layer.

Neural Simplex architecture [83] shares some features with SR. However, Neural Simplex is applied to low-level continuous control only, while in this work SR is applied to perception with a path to end-to-end verifiable safety for complex CPS. Neural Simplex's cross-layer interaction is limited to aiding improvements in the learned behaviors of a reinforcement learning controller, as compared to the varied synergistic co-operations in SR. SR also allows the safety layer to cover specific faults of the mission layer rather than complete functionality, enabling incremental integration.

Safe Autonomous Driving has been an active area of research with varying approaches. System architectures have been proposed that improve security and execution time predictability of parts of the mission software [84]–[86]. While this greatly decreases the response latency variance for the AV, it does not address faults born out of complex algorithms and software pipelines. Techniques to improve driving policies [87], path planning [88]–[90], risk assessment [91]–[95], steering or braking control [96]–[100] are still vulnerable to perception failures. SR framework addresses the perception failures first and is able to leverage the above advancements in path planning and control by allowing their use in the mission layer and even potentially incorporating them into the safety layer if they meet the safety layer requirements.

Sensor Fusion is a popular form of functional redundancy used in AV [101], [102] to address faults in obstacle detection. Using different sensors for the shared function of perception helps create a combined output that does not suffer from an individual sensor's limitations. As an example, objects not distinguishable by visual perception using a camera input may still be detected by radar. However, the intermingling of safety and performance optimizations and limitations of algorithms used to interpret sensor inputs have resulted in lapses in perception and collisions [12]–[15], [41], [45]. Unforeseen interactions between features unnecessary for safety and optimizations for performance have led to crashes [13]. SR safety layer algorithms have clear constraints based on the sensor and algorithm properties. Sensor fusion can be used to systematically cover different constraints enabling higher availability of verified capabilities in adverse conditions.

Certified Control [103] is another approach to achieving end-to-end verifiable safety in AV. While based on some shared premise, *i.e.,* unverifiability of DNN, small trusted based and runtime monitoring, we assert that SR has specific advantages over certified control. Certified control, based on checking of proposed action vs sensor data, is necessarily post-processing. In comparison, the SR safety layer runs parallel to the mission layer enabling cooperation and synergies. In the specific case of perception, the fast algorithm employed in this work can reduce the workload for the mission layer and provide fault information before subsequent tasks like path planning have been completed. A benefit of certified control over the presented design of SR is that the SR safety layer overrides the mission layer actions when the mission layer exhibits perception faults that pose a risk to the safe operation of AV, where certified control would only do so when the fault affects the proposed action of the AV. However, similarly unnecessary overrides may happen in certified control when there is an issue in certificate creation only. These overrides are only a performance issue and do not impact the safety of the AV. Admittedly, certificate checking can be integrated into SR to expand the fault tolerance. However, where available, SR's method of verifiable fulfillment of minimal safety requirements will provide superior system performance and safe actions available via the safety layer alone.

Summary. SR integrates the design philosophies of various prior safety architectures for AV while providing the flexibility of incremental addition of capabilities making it practical and easy to include in existing AV software stacks. Many existing approaches for safe control can be housed within this architecture to expand the capabilities and fault handling of the safety layer. This work provides a path to the integration of various safety techniques in AV achieving a result that is better than the sum of its parts.

III. Synergistic Redundancy

Machine learning has enabled applications that might have remained impossible otherwise. The complexity of the tasks involved in such applications has made achieving similar capabilities and performance with classical algorithms, as yet, impossible. However, classical algorithms can fulfill specific safety requirements for these tasks while being fully verifiable for their behavior, as shown by our prior work [25]. SR pairs machine learning fulfilled tasks with corresponding classical algorithms that provide specific safety guarantees while improving upon both by enabling synergistic cooperations. We first discuss the general model of SR architecture before exploring how it can be applied toward resolving perception faults in autonomous vehicles (AV). The defining components of SR, as shown in Figure 1, are:

- Mission Layer (***Mission***): The mission layer is a complex and high-performance agent capable of and responsible for fulfilling the mission of the system, *e.g.,* driving an AV from a location to a destination.
- Safety Layer: The safety layer is responsible for providing specific safety guarantees. The safety layer must satisfy the verification requirements for safety-critical software, *i.e.,* be logically analyzable and fully verifiable [30], [31]. The safety layer is not required to be able to fulfill the mission of the system but rather only to be able to reach a safe state. It is functionally divided into two parts,
  (a) Algorithms (***SafeAlg***) that provide specific capabilities, *e.g.,* obstacle existence detection in AV;
  (b) Fault Handlers (***FaultHandler***) that leverage the algorithms to detect and correct specific faults, *e.g.,* false negatives in object detection.

With the mission and safety layers the system is able to fulfill its purpose while maintaining safety. We now describe the system while defining requirements (***Req***) and optional features (***Optional***) for the general design of SR architecture.

A. Isolation

One of the features of the safety and mission separation design is the protection of the safety layer from system faults, *e.g.,* software crashes or panics, and temporal perturbations of the mission layer. This is achievable via three methods

- Platform: The layers run on different hardware platforms.
- Baremetal: The layers run on the same platform but on different core clusters, so the software stacks for the layers are completely isolated.
- Hypervisor: The layers are isolated with type-1 baremetal virtualization or type-2 hosted virtualization.

Additionally, such separation makes temporal predictability easier to achieve for the safety layer, isolating any perturbations from the mission layer. While this is an intrinsic feature of the platform and baremetal separation, real-time scheduling support for hypervisors has been developed in prior works [104]. As detailed in Section II such isolation is established technology, which we leverage in SR architecture.

B. Mission Layer

In SR, scarce few requirements are imposed on the mission layer, enabling independent development of mission layer components. The mission layer represents a high performance complex autonomous system. In case of faults in the mission layer, the continued fulfillment of the mission of the system is not guaranteed. The mission capability of the CPS is completely dependent on the mission layer. Hence, the mission layer should be independently tested to be able to function in common cases with reasonable confidence.

Mission-Req-1: Expose internal states and intermediate outputs to safety layer (§IV-B1). The specific interfaces required are dependent on components of the safety layer. This is easily achieved in publisher-subscriber models of communication. State-of-the-art open source AV software, Apollo [53] and Autoware [105], meet this requirement.

Mission-Optional-1: Utilize optimization hints from safety layer (§IV-B2).

Mission-Optional-2: Utilize faults detected by safety layer to correct mission layer behavior (§IV-B3).

Mission-Optional-3: Provide interfaces to capabilities, usable by the safety layer to request specific corrective actions, too complex to be fully executed by the safety layer (§IV-B4).

C. Safety Layer

The safety layer hosts only verifiable and logically analyzable software. Since simplicity is a design goal for the safety layer

1) Safety Algorithms: Safety algorithms fulfill tasks that contribute to system safety and provide specific guarantees.

SafeAlg-Req-1: A list of *capabilities* (§IV-C1). This also includes any interfaces for required inputs and primary outputs.

SafeAlg-Req-2: A list of *commitments* or guarantees that are provided by the algorithms(§IV-C2).

SafeAlg-Req-3: A list of *constraints* under which the commitments are valid (§IV-C3).

2) Fault Handlers: Fault handlers must determine whether safety and mission layer outputs are valid and initiate corrective actions when faults are detected. Fault handling can leverage other layers' capabilities as long as faults are not detected within them.

FaultHandler-Req-1: Ability to monitor adherence to safety layer constraints (§IV-D1). In case of a constraint violation, corrective actions to either restore adherence to the constraint or handle the loss of safety layer capability.

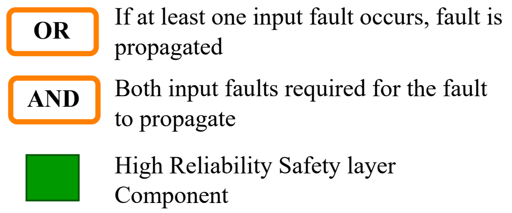
FaultHandler-Req-2: For each capability in safety layer, a method for fault detection, comparing mission and safety outputs for the same functionality (§IV-D2).

FaultHandler-Req-3: For each fault that can be detected, a logic for determining the criticality of the fault (§IV-D3).

FaultHandler-Req-4: Response to each fault that may be detected (§IV-D4). Response to safety-critical faults must be executable by verifiable software in safety, decision, and control components. Lower criticality fault resolution can also involve unverifiable software *e.g.,* DNN in the mission layer.

TABLE I: Summary of Synergistic Interactions

Mission Layer ← ***Safety Layer***	
IV-B2, IV-C1	Optimize execution by work reduction and prioritization.
IV-B3, IV-D2	Fault information for correction and online learning.
Safety Layer ← ***Mission Layer***	
IV-D4	Safe stop when safety algorithm constraints are violated but no safety-critical faults were detected.


OR
If at least one input fault occurs, fault is propagated
AND
Both input faults required for the fault to propagate
High Reliability Safety layer Component
Fig. 2: Gates and color code used in the fault trees in Figure 3.

IV. Reference Design

The previous section established the general structure of *SR*. We now describe the application of *SR* to address obstacle existence detection faults. As noted earlier, such faults have been involved in real-world collisions. Specifically, we describe how each requirement and optional feature of *SR* is fulfilled and leads to the handling of these faults. Furthermore, Table I summarizes synergistic interactions between the layers.

A. Scope and Assumptions

Obstacle existence detection fault or False Negative (FN) refers to a condition where obstacles in the AV's vicinity are not perceived to exist. In a prior work [25] we surveyed all autonomy-involved traffic collision fatalities and found that in each case obstacle existence detection faults were determined or suspected. The complete fault model used in this work is elaborated with the help of fault tree analysis (FTA) [106]. Specifically, we show how the usage of *SR* for handling obstacle existence faults changes the fault trees for AV, using prior FTA for AV as base [107], [108]. Figure 2 describes the components of the fault tree. High-reliability components are composed of verifiable software only. Faults in high-reliability components, colored green, have an exceedingly low probability of occurring when their preconditions are being met. This high confidence is based on their analyzable logic, fully verifiable implementations, and runtime monitoring of constraints. Figure 3 shows the specific faults addressed in the presented design.

Each fault in Figure 3 that is not mitigated by a high-reliability capability in the safety layer is an avenue for future research. *SR* provides a flexible architecture where all such future capabilities can be integrated, incrementally adding to the safety guarantees.

We assume that all communication is real-time and lossless. Data shared between layers is assumed to be always fresh and valid for its use. Temporal safety *i.e.,* real-time execution and communication is an orthogonal concern to the functional safety addressed in this work and has been the focus of prior works [86], [109]–[111].

B. Mission Layer

The mission layer is an autonomous driving agent capable of fulfilling the mission of the AV, i.e. driving from one point to another while maintaining safety for most common scenarios. However, if the safety layer detects a reason to override the mission layer, the mission of the AV is abandoned till the mission layer faults are mitigated. In this work, Baidu's Apollo 5.0 [53] is used as the mission layer.

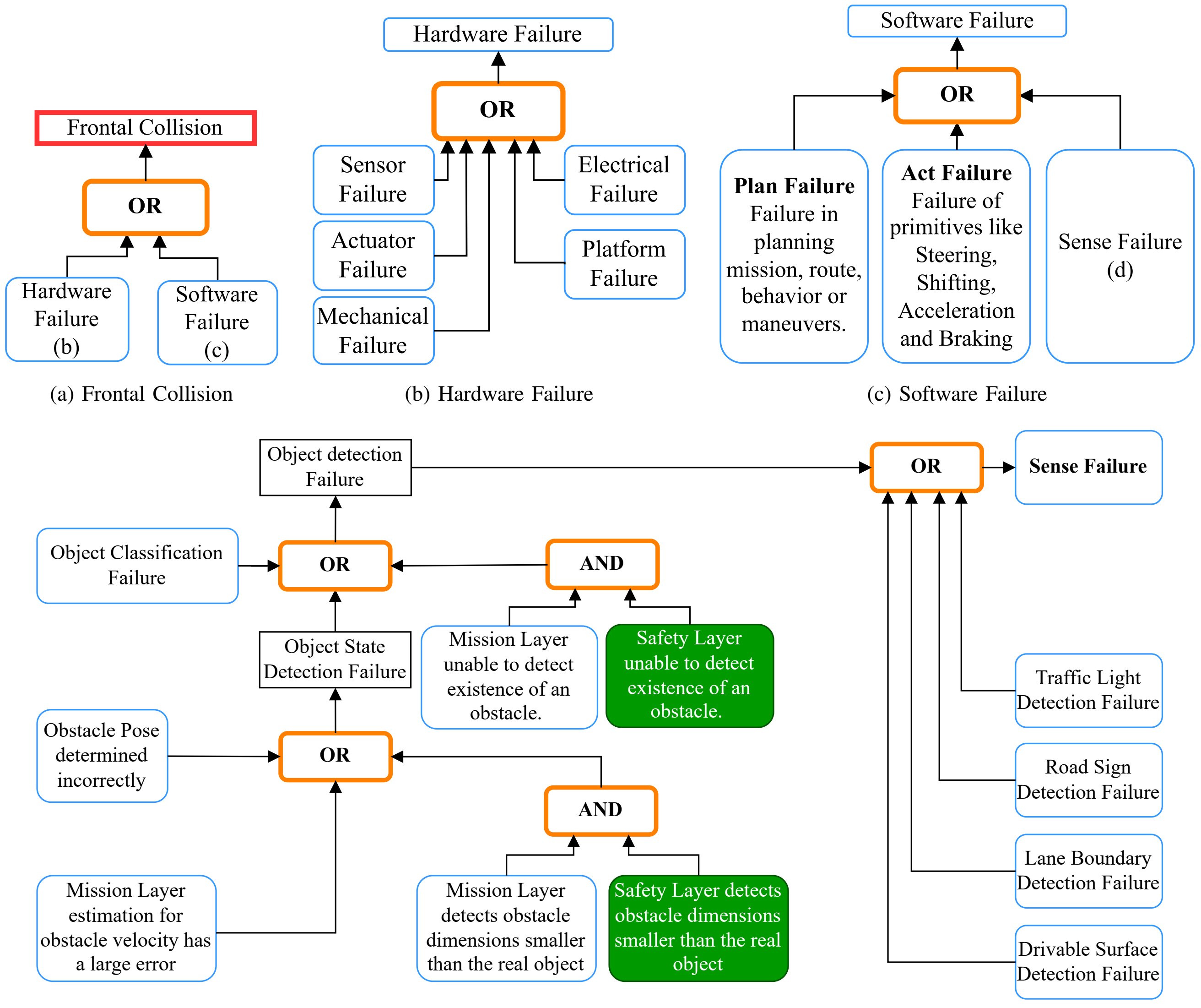
The clear separation of the mission layer from the safety layer makes the *SR* framework forward compatible with all improvements in the autonomous driving agents. While in this work we do propose synergies between the two layers and optimization opportunities in the mission layer, the changes in the mission layer are not necessary for the safety guarantees (§III-B ***Mission-Optional-1-3***).

1) ***Mission-Req-1****:* The only requirement is for the mission layer to expose its perception information to the safety layer, which in the case of Apollo CyberRT [112] is easily achieved by subscribing to these outputs.

2) ***Mission-Optional-1****:* Since the safety layer executes low latency verifiable obstacle detection algorithms, it can provide this output to the mission layer's object detection, identifying regions in the vicinity of the AV where obstacles exist and the risk of collision posed by such obstacles. Recent works have shown that such information can be used to enable significant optimizations in mission-critical systems, *e.g.,* mission layer in *SR*. These optimizations reduce execution workload, latency, and resource requirements of the mission-critical object detection DNN. They also enable careful prioritization of higher-risk objects in the AV's environment within the mission-critical object detection pipeline.

Liu *et al.* [113], [114], Hu *et al.* [115], [116] and Chen *et al.* [117] address the priority inversion inherent to DNN. An external agent slices the input camera images into regions and allocates priority to each region. The priorities are then utilized by the machine intelligence software to prioritize the processing of higher criticality regions. In AV, such priority inversions exist in perception, wherein detection of higher criticality objects in a scene suffers delays due to the existence of lower criticality objects in the same frame. Kang *et al.* [118] and Liu *et al.* [119] remove the use of external region proposals. Instead, constant regions [118] or predictive regions based on prior full frame detection and pixel flows between frames [119] are used.

At this point, a natural question occurs, why don't we add a low latency obstacle detection logic to existing object detection DNN and reap the performance benefits without the comparatively complicated separation into mission and safety layers? Such an approach would allow the benefits shown in these works [113]–[117], but would compromise safety. In absence of verifiable guarantees, external region proposal


Frontal Collision
OR
Hardware Failure (b)
Software Failure (c)
Hardware Failure
OR
Sensor Failure
Actuator Failure
Mechanical Failure
Electrical Failure
Platform Failure
Software Failure
OR
Plan Failure
Failure in planning mission, route, behavior or maneuvers.
Act Failure
Failure of primitives like Steering, Shifting, Acceleration and Braking
Sense Failure (d)
Object detection Failure
OR
Sense Failure
Object Classification Failure
OR
AND
Object State Detection Failure
Mission Layer unable to detect existence of an obstacle.
Safety Layer unable to detect existence of an obstacle.
Obstacle Pose determined incorrectly
OR
AND
Mission Layer estimation for obstacle velocity has a large error
Mission Layer detects obstacle dimensions smaller than the real object
Safety Layer detects obstacle dimensions smaller than the real object
Traffic Light Detection Failure
Road Sign Detection Failure
Lane Boundary Detection Failure
Drivable Surface Detection Failure

(a) Frontal Collision

(b) Hardware Failure

(c) Software Failure

(d) Sense Failure

Fig. 3: Fault tree for AV [107], [108] modified by the inclusion of the safety layer. Figure 2 describes the gates used. High-reliability safety layer components inhibit the propagation of corresponding faults by the mission layer.

agents may erroneously exclude safety critical obstacles. Fixed regions as chosen by Kang *et al.* [118] accommodate the common region of interest for driving but cannot account for the complex dynamic nature of collision risk in AV potentially delaying detection of critical obstacles. Self Cued region proposal [119] has reduced mAP, even when considering critical objects only, compared to contiguous full frame processing, due to accumulation of uncertainty between inspections and inaccuracies in the optical flow estimations. These potential safety-critical faults are in addition to any algorithmic faults that already exist in the base object detector DNN.

The safety layer designed in this work, utilizing verifiable obstacle existence detection algorithms [25], would provide deterministic guarantees of including all high criticality obstacles within the proposed regions. Mission layer object detection benefits from reliable region proposals from the safety layer, optimizing its execution, in turn providing detection output for critical objects faster to any users of this output in both mission and safety layers.

The separation of safety vs. mission layers also contributes to the safe usage of these optimizations. Since the safety layer can detect and respond to safety-critical obstacles, even when the mission layer does not perceive their existence, *i.e.,* in *SR* any inadvertent FN errors caused by such optimizations affect only the mission of the AV, not its safety.

*3) **Mission-Optional-2**:* Information about any non-critical FN in mission layer output, detected in safety layer, can be used by the mission layer object detector. It can prioritize the region of FN, dedicate extra computation or process higher resolution input for the region.

*4) **Mission-Optional-3**:* When safety layer constraints are violated (§IV-D1), but no critical faults have been detected prior to these violations, the AV can enter a degraded state. In the absence of actual FN errors, emergency braking is an unnecessarily extreme reaction. Instead, the safety layer can change the mission of the AV, requesting a safe stop, *e.g.,* pull over to the side of the road.

C. Safety Layer - Algorithms

The algorithms used in the safety layer provide specific capabilities and commitments, when their required constraints are met. To address faults in obstacle existence detection we use Depth Clustering, a LiDAR-based geometric algorithm, developed by Bogoslavskyi and Stachniss [54], [55] and analyzed in a prior work [25] to establish a detectability model *i.e.,* a deterministic model for when obstacles are detected. This detectability model is used to determine the properties below:

*1) **SafeAlg-Req-1**:* Capability to detect the existence of obstacles, using LiDAR range images as input and provide bounding boxes for obstacles. This output is made available to the mission layer where it is usable to improve the object detection pipeline, as discussed in Section IV-B2.

*2) **SafeAlg-Req-2**:* From our prior work [25], using parameters from the default LGSVL LiDAR sensor [120], the following detectability model can be derived. Where y is obstacle height and x is the distance from the LiDAR sensor to the obstacle, the obstacle is always detected when the following inequality holds:

$$y \geq 0.037x - 0.034. \tag{1}$$

While the exact model and reasoning are established in this prior work [25], intuitively this model implies that given the height of an obstacle, it is always detected when closer than a specific limit. Note that the reverse is not necessarily true, objects may still be detected when further away from this limit, however, the guarantees no longer apply. The constants in (1) are dependent on LiDAR and algorithm parameters, *e.g.,* all else being equal including the field of view, using a 64 beam LiDAR will yield a longer detection range guarantee than a LiDAR with 32 beams.

Let's denote the maximum range of the LiDAR sensor in clear weather by $R^{LiDAR}_{max,clear}$. This is a sensor specification, *e.g.,* 75 meters for top LiDAR in Waymo Open Dataset [121]. Under clear weather conditions, this is the maximum detection range of LiDAR, *i.e.,*

$$R^{LiDAR}_{max} = R^{LiDAR}_{max,clear}. \tag{2}$$

Let's further assume that there is a safety standard requiring AV on the road to be able to detect obstacles till a minimum height of H^O. Using (1) the maximum range at which this object of height H^O can be detected can be determined. Let's denote this range as $R^O_{max} = \frac{H^O + 0.034}{0.037}$. The maximum range at which obstacles can be reliably detected (R_{max}) is then

$$R_{max} = min(R^O_{max}, R^{LiDAR}_{max}). \tag{3}$$

Equation (3) establishes the maximum detection range. However, as a safety policy and to accommodate for any error allowances within the system, a safety margin can be added to this range, denoted as D^{Safety}. Combining this with (3) we get the maximum stopping distance for the AV:

$$D^{stop}_{max} = R_{max} - D^{Safety}. \tag{4}$$

Let's further assume that the worst-case computational latency of the AV software stack *i.e.,* time delay between sensor input to actuation commands being asserted is L_{max}, and the maximum safe deceleration of the AV is a^{av}_{max}. The maximum safe operating speed of the AV is then derived by solving:

$$D^{stop}_{max} = v^{safe}_{max}(t + L_{max}) - \frac{1}{2}a^{av}_{max}t^2 \tag{5}$$

$$v^{safe}_{max} = a^{av}_{max}t \tag{6}$$

where t is deceleration time. By substituting t in the first equation with $t = \frac{v^{safe}_{max}}{a^{av}_{max}}$ found in the second equation, we have

$$v^{safe}_{max} = -a^{av}_{max}L_{max} + \sqrt{(a^{av}_{max}L_{max})^2 + 2a^{av}_{max}D^{stop}_{max}}. \tag{7}$$

*3) **SafeAlg-Req-3**:* The detectability model is valid under three constraints, as below [25]. Example values are based on LiDAR parameters from Waymo Open Dataset [121].

- obstacles must meet a minimum width requirement, *i.e.,* horizontal separation between LiDAR beams.
- obstacles are further than first LiDAR return from the ground more than 6.7 m away from the AV, which is mitigated by leveraging additional low height close range LiDAR as present in the dataset or proximity sensors.
- LiDAR beams must return from the first obstacle they encounter, implying that there are no returns lost due to energy dissipation and there are no returns closer than actual obstacles. Weather impediments like fog or rain can violate this constraint, therefore, we discuss the handling of this constraint in Section IV-D1.

D. Safety Layer - Fault Handlers

*1) **FaultHandler-Req-1**:* As noted in Section IV-C the primary dynamic constraint of the obstacle existence detection that needs to be monitored is the validity of LiDAR returns. This constraint can be violated when particulate impediments in air, *e.g.,* smoke, dust, or fog, reflect and dissipate LiDAR beam energy, causing false early or lost returns [122]–[125]. The detection of these conditions is an orthogonal problem with significant prior solutions [126]–[130]. Note that we only consider benign faults, not LiDAR spoofing attacks [131]–[134], which can also violate this constraint.

The first solution is velocity limitation to ensure that the obstacle detection guarantees cover the safe stopping distance of the AV. The power of LiDAR attenuates per Beers-Lambert Law [135]:

$$\tau(R) = \frac{P(R)}{P(0)} = e^{-\sigma R} \tag{8}$$

where $P(R)$, $P(0)$ refer LiDAR power at distance R and 0, respectively. The value σ is the attenuation coefficient that characterizes how strongly a medium absolves the wave signals. The attenuation coefficient is calculated by the Kruse model [136]

$$\sigma(\lambda) = \frac{17.35}{V}(\frac{\lambda}{0.55})^{-q}$$

where V is the visibility which depends on air quality and condition, such as fog and haze. The parameter λ is the wavelength of the signals in the micrometer. A typical LiDAR wavelength is either 905 nm or 1550 nm. The parameter q depends on visibility [137] as

$$q = \begin{cases} 1.6 & V > 50\ km \\ 1.3 & 6\ km < V < 50\ km \\ 0.16V + 0.34 & 1\ km < V < 6\ km \\ V - 0.5 & 0.5\ km < V < 1\ km \\ 0 & V < 0.5\ km \end{cases}$$

Knowing the LiDAR's visibility under the current weather condition is critical because the short-sighted perception will delay the vehicle's response and limit the ability to respond to obstacles. We emphasize this limitation but leave the accurate calculation of the measurable range as future work because this is out of the current paper's scope and has been studied in prior works [138]. Instead, we provide a rule of thumb as follows.

Lemma 1. Assume $R^{LiDAR}_{max,current} \leq R^{O}_{max}$, where $R^{LiDAR}_{max,current}$ is the maximum range of the LiDAR under the current weather condition. Given nominal maximum measurable distance of LiDAR R_{max} and attenuation coefficient σ_{clear} under clear weather, one can find the maximum safe velocity as follows:

$$v^{safe}_{max} = -a^{av}_{max}L_{max} + \sqrt{(a^{av}_{max}L_{max})^2 + 2a^{av}_{max}(\frac{\sigma_{clear}}{\sigma_{current}}R^{LiDAR}_{max,clear} - D^{Safety})}. \tag{9}$$

Proof. Regardless of the weather condition, minimum observable signal power remains identical to the same LiDAR device, *i.e.,* there exists τ_{min} in (8). Therefore, we have

$$\tau_{min} = e^{-\sigma_{clear}R^{LiDAR}_{max,clear}} = e^{-\sigma_{current}R^{LiDAR}_{max,current}}$$

for clear weather and current weather. By comparing the exponent, we have the maximum measurable distance of LiDAR under the current weather as follows:

$$R^{LiDAR}_{max,current} = \frac{\sigma_{clear}}{\sigma_{current}}R^{LiDAR}_{max,clear}. \tag{10}$$

By definition, $\sigma_{clear} \leq \sigma_{current}$. Combining (2) and (10) with the above observation, during current weather conditions we get

$$R^{LiDAR}_{max} = R^{LiDAR}_{max,current} \leq R^{LiDAR}_{max,clear}. \tag{11}$$

The assumption $R^{LiDAR}_{max,current} \leq R^{O}_{max}$, (3) and (11) imply that:

$$R_{max} = R^{LiDAR}_{max,current} \tag{12}$$

By plugging (12) into (4) and (7), we have (9). □

Lemma 1 provides an intuitive rule of thumb about maximum velocity change under different weather conditions. Assume the computational latency is small enough (*i.e.,* $L_{max} \simeq 0$), and ignore the safety margin (*i.e.,* $D^{Safety} \simeq 0$). Then, the maximum safe velocity in (9) is found by

$$v^{safe}_{max} = \sqrt{\frac{\sigma_{clear}}{\sigma_{current}}}\sqrt{2a^{av}_{max}R^{LiDAR}_{max,clear}}$$

Given the fact that the typical attenuation coefficients are 0.1 for clear air, 1 for haze, and 10 for fog [137], the vehicle should reduce its maximum speed to $\sqrt{0.1/10} = 1/10$ for dense fog and $\sqrt{0.1/1} \simeq 1/3$ for haze, compared to the safe speed limit under clear weather, $\sqrt{2a^{av}_{max}R^{LiDAR}_{max,clear}}$ if $R^{LiDAR}_{max,clear} \leq R^{O}_{max}$. On the other hand, if $R^{LiDAR}_{max,clear} > R^{O}_{max}$, then we have

$$v^{safe}_{max} > \sqrt{\frac{\sigma_{clear}}{\sigma_{current}}}\sqrt{2a^{av}_{max}R^{O}_{max}}.$$

Therefore, reducing speed limit to $\sqrt{0.1/10} = 1/10$ for dense fog and $\sqrt{0.1/1} \simeq 1/3$ for haze, compared to the safe speed limit under clear weather, $\sqrt{2a^{av}_{max}R^{O}_{max}}$, guarantees safety as well. It is worth emphasizing that this calculation is valid for the maximum safe velocity limit, *i.e.,* one does not need to reduce the speed further if its original speed is already within the adjusted safe velocity range. It is also notable that the braking deceleration a_{av} in Lemma 1 can change under adverse weather conditions, *e.g.,* due to icy roads, reducing the maximum safe velocity further.

On the other hand, adverse weather can result in false positive reflections of Lidar, which can trigger the safety layer to conduct emergency stops. Reducing speed may help avoid emergencies by allowing the mission layers to deal with more cases. False positive can degrade the performance but is not safety critical. Furthermore, false positive returns occur in very close proximity, *i.e.,* within one ft from the Lidar, as shown in Fig. 5 in [122], where catastrophic weather has been studied with visibility between 13 m and 45 m.

2) ***FaultHandler-Req-2:*** To determine obstacle existence faults in the mission layer's object detection output, it is compared against the obstacle detection output from the safety layer algorithm. The comparison uses the minimized requirements for obstacle detection, as established in the prior work [25]. Intuitively, using the minimized requirements, we evaluate whether all the obstacles detected by the safety layer algorithm are adequately detected by the mission layer, and if any obstacle is not, a fault is detected.

3) ***FaultHandler-Req-3:*** To determine the appropriate reaction to a fault, first, its severity must be established. For any obstacle existence detection fault, it must be ascertained if this obstacle poses a risk for collision with the AV. We use a physical model for collision risk from our prior works [139], wherein an obstacle is determined to pose a risk of collision if their existence regions [140] overlap within the time to stop the AV. A fault involving an obstacle that poses a risk of collision

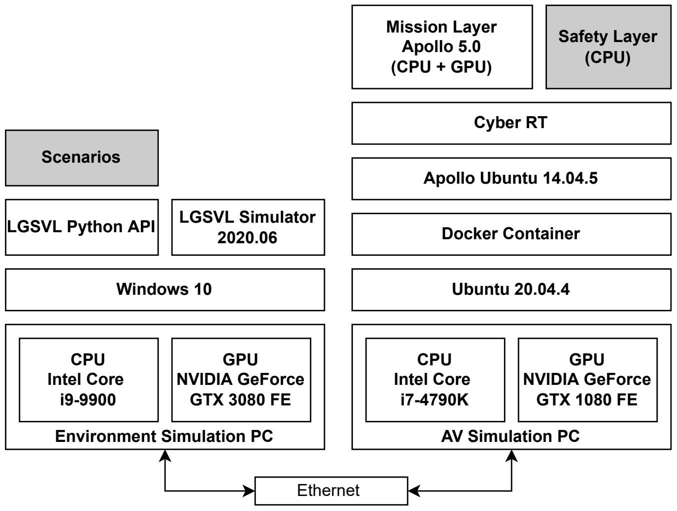


Fig. 4: Simulation setup for evaluations showing hardware and software stack. The shaded components were developed by the authors for this evaluation, except the Depth Clustering [54], [55] obstacle detection algorithm in safety layer. The mission and safety layers are as shown in Figure 5.

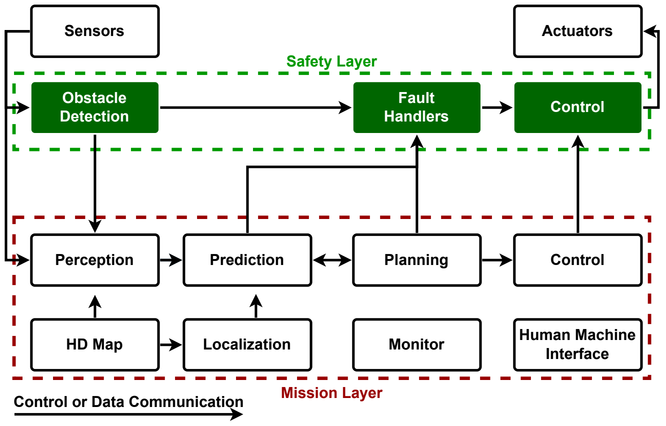


Fig. 5: Control and Data Pipeline of an Autonomous Driving system within Synergistic Redundancy framework. The mission layer uses Baidu's Apollo 5.0 Driving Agent [141].

is safety-critical. It should be noted that any deterministic and worst case conservative model for risk can be substituted here.

*4) **FaultHandler-Req-4**:* There are several actions imposed based on operations conditions and faults detected.

a) Velocity Limitation: To accommodate for algorithmic constraint (§IV-C), velocity is limited to maintain the validity of obstacle existence detection algorithm, as described in (7) and (9), .

b) Safety-critical Obstacle existence detection fault: An emergency brake is applied to avoid the potential collision.

c) Non-critical Obstacle existence detection fault: Fault information is made available to the mission layer but no action is initiated.

d) Mission Abandon: When physical impediments become too dense for the validity of the obstacle detection algorithm to be maintained with velocity limitation, but no obstacle existence detection faults have been determined, The safety layer can change the mission of the AV, directing it to reach a safe stopping position.

V. Evaluation

We evaluate the efficacy of the SR system design by showcasing its behavior with a simulation. Figure 4 shows an overview of the simulation setup and Figure 5 shows an overview of the layer components.

- *Environment*: LGSVL [57] environmental simulator controlled with LGSVL Python APIs [142].
- *Mission Layer*: This layer is Apollo 5.0 [53] Driving agent, from its LGSVL fork [56]. We route control commands generated by Apollo through the safety layer control component, as shown in Figure 5.
- *Safety Layer*: Safety layer includes obstacle existence detection using Depth Clustering [54], [55] (§IV-C), and the fault handling logic (§IV-D). The safety layer runs on the same machine as Mission Layer for this evaluation. Isolation between the layers with resultant computational and communication implications will be addressed in future work.

For this evaluation we setup the autonomous agent in 4 different configurations:

- **Mission**: Apollo 5.0 *i.e.,* mission layer controls the AV. The safety layer is disabled. This scenario provides the baseline for the mission capabilities of the system.
- SR (**Mission Crash**): Mission layer becomes unavailable at $t = 0$. Hence safety layer would apply an emergency brake as the simulation starts and the braking will become effective after the control delay. Similar functionality is also provided by Apollo's Guardian module. This case represents an ideal emergency braking response and provides the synthetic ideal best case for safety.
- SR (**Fault Injected**): Mission layer is available and its control is routed via the safety layer. However, within the safety layer's fault handlers, obstacle existence detection faults are synthetically injected, causing the safety layer to start braking when the obstacle is detected. Safe handling of obstacle existence detection faults is the primary focus of this work (§IV-A), however, the causes of such faults are beyond the scope.
- SR: The SR system with no instrumentation.

Due to their varied capabilities and responsibilities, the intention here is not to compare the mission and safety layer behaviors, rather **Mission** setup provides a baseline for the mission behavior and SR (**Mission Crash**) setup provides a best case response, under the dynamics of the scenario and that of the simulation vehicle Lincoln MKZ 2017 [143]. Finally, using simulation parameters, $R^{LiDAR}_{max,clear} = 100\ m$, $D^{Safety} = 0.1\ m$, $a^{av}_{max} = 7.5\ m/s^2$, $L_{max} = 0.01\ s$, and $H_O = 0.75\ m$ for the back of a sedan with (7) we get $v^{safe}_{max} = 17.71\ m/s$.

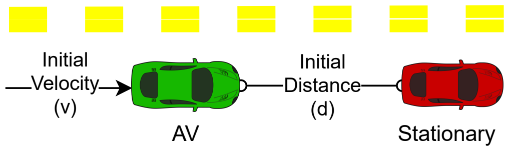


Fig. 6: Evaluation scenario where the AV approaches a stationary obstacle. The AV is moving with some initial velocity v when at a distance d away from the obstacle. We let various configurations of *SR* control the AV from this point (§V-A). Each such run leads to either a collision between the AV and the obstacle, AV stopping safely before the obstacle or the AV safety overtaking the obstacle.

TABLE II: Outcomes

Requirement	Collision	Safe Stop	Safe Pass
Safety	Violated ✗	Met ✓	Met ✓
Mission	Violated ✗	Abandoned ✗	Met ✓

A. Scenario

Let's consider a synthetic scenario as shown in Figure 6. The scenario starts with the AV moving with some initial velocity, $v(0) \in \{5, 10, 15, 20, 25, 30, 35, 40\}$ m/s. Another vehicle is placed in the AV's path at an initial distance, $d(0) \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$ m. For all times $t > 0$ only the autonomous agent stack controls the AV.

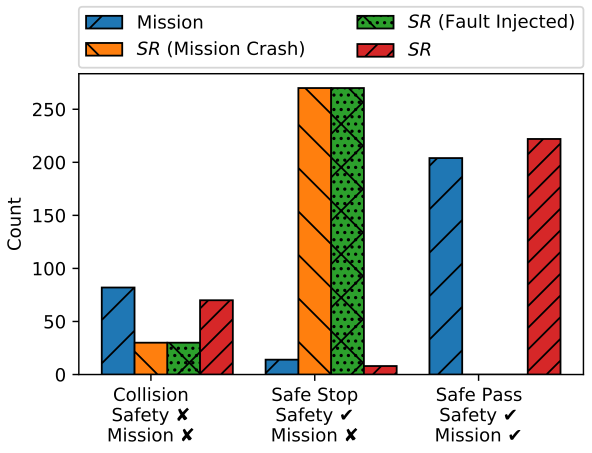
The AV is given the mission to pass the obstacle without colliding with it, *i.e.,* AV's mission is to both pass the obstacle and avoid collisions. The safety requirement during this mission is to avoid any collisions.

B. Result Overview

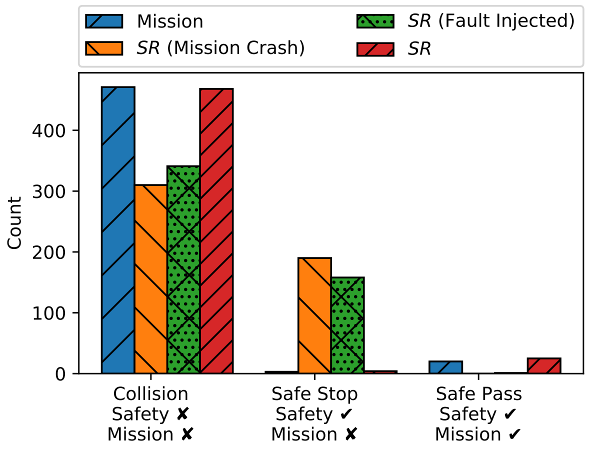
The simulation yields the following outcomes, as also summarized in Table II:

- *Collision* x : safety requirement is violated and mission of the AV cannot be fulfilled
- *Safe Stop* ● : safety requirement of no collisions is maintained but the mission of AV could not be fulfilled.
- *Safe Pass* + : safety requirement and mission are fulfilled.
- *Invalid*: In some instances, vehicle control becomes unstable, leading to the vehicle going out of the road boundaries. This occurs most commonly at high initial velocities. Such instances are marked invalid and their results are not included in the rest of this section. For each $v(0)$ $d(0)$ combination, among the 10 repetitions, at least one valid result was recorded.

We now compare the prevalence of these outcomes, among the different autonomous agent configurations. Figure 7a shows the outcomes when initial velocity is below the safe limit, *i.e.,* $v(0) \leq v^{safe}_{max}$ and Figure 7b shows the outcomes when initial velocity is above the safe limit, *i.e.,* $v(0) > v^{safe}_{max}$. In each figure, the count (Y-Axis) is the number of instances out of the total 800 that lead to the specific outcome (X-Axis), excluding Invalid.



(a) $v \leq v^{safe}_{max}$

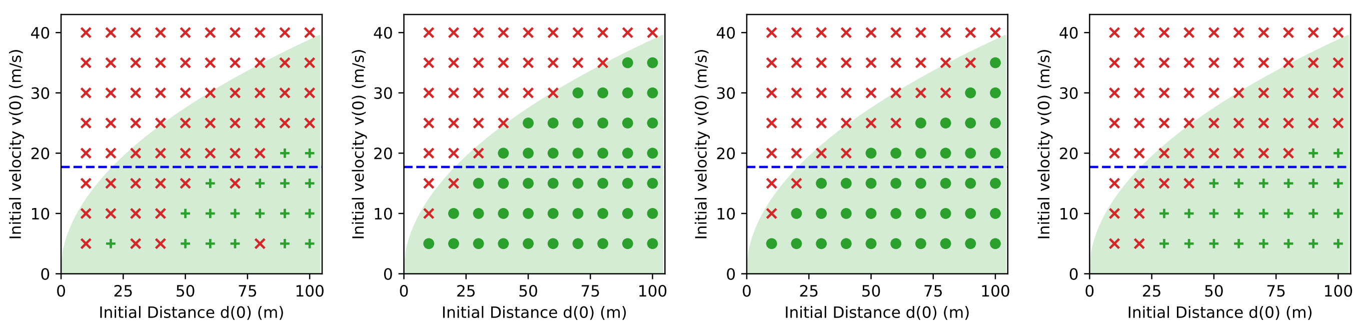


(b) $v > v^{safe}_{max}$

Fig. 7: Comparison of scenario outcomes. The X-Axis shows the three possible outcomes, *i.e.,* Collision, Safe Stop, and Safe Pass, summarized in Table II. The Y-Axis counts how often, among all the simulation instances for each configuration, the specific outcome was observed. The scenarios are described in Section V-A.

Discussion. When $v \leq v^{safe}_{max}$ *SR (Fault Injected)* behaves identical to the best case configuration *SR (Mission Crash)*. This observation supports the primary claim of this work, *i.e.,* deterministic safety limits and collision avoidance when faced with obstacle existence detection faults.

However, when the mission layer detects the obstacle but is unable to avoid the collision, by design the safety layer does not intervene. With the capability of obstacle existence detection (§IV-C1), the safety layer can only ensure that the mission layer's perception detects the obstacle. Due to the lack of planning and control capabilities in the presented design, the safety layer here cannot determine if given the correct perception input mission layer has not chosen the optimal path.



(a) Mission (b) *SR* (Mission Crash) (c) *SR* (Fault Injected) (d) *SR*

Fig. 8: Each data point here represents the worst case outcome of the 10 repetitions conducted for each configuration and combinations of $d(0)$ (X-Axis), $v(0)$ (Y-Axis). The shaded green area, defined by $v(0) \leq \sqrt{2 * 7.5 * d(0)}$, represents the distance and velocity combinations at which it is physically possible for the AV to avoid a collision, assuming constant deceleration of 7.5 m/s^2 and zero response latency. x represents that a collision occurred, ● denotes that no collisions were observed and + denotes the AV was able to pass the obstacle. The blue dashed line notes $v_{max}^{safe} = 17.71\ m/s$.

When $v > v_{max}^{safe}$ the max range at which the obstacle is detected by the safety layer obstacle detection algorithm (R_{max}^{O}) is not guaranteed to be a sufficient distance for the AV to brake to stop. Safe stops still happen because

(i) although not guaranteed, due to the discrete nature of LiDAR beams obstacles may be detected beyond R_{max}^{O};

(ii) false positives cause unnecessary early braking which in this case creates additional slack time for collision avoidance;

(iii) when $d(0)$ is large, the mission layer's actuation commands reduce AV speed to match its planning and control goals, helping avoid collisions.

Finally, the safety layer in *SR* does not share the mission layer's goal of completing the AV's mission. Hence, when mission layer has any faults, *SR (Mission Crash)* and *SR (Fault Injected)*, the mission of the AV, *Safe Pass*, is abandoned.

C. Detailed Results

We now discuss the results for this scenario in detail with scatter plots, as shown in Figure 8. Among all valid outcomes for each $v(0), d(0)$ combination if any instance results in a collision, the result notes that data point as a *Collision* (x). Only when all repetitions are able to safely pass the obstacle the noted result is a *Safe Pass* (+). When no collisions occur and at least one instance does not lead to a safe pass, the result is noted to be *Safe Stop* (●).

$v_{max}^{safe} = 17.71\ m/s$ is indicated by blue horizontal lines in each plot. The shaded area ($v(0) \leq \sqrt{2 * 7.5 * d(0)}$) notes the best possible braking response, assuming $a_{max}^{av} = 7.5\ m/s^2$. This ignores any computation latency or dynamics of the simulation. Figure 8 (b) is the synthetic best case for safety where the AV applies maximum brake at the start of the simulation run. Figure 8 (a) is the mission baseline.

Discussion. Figure 8 (c) captures the *SR* behavior under obstacle existence detection faults, the focus of this work (§IV-A). This result substantiates the $v_{max}^{safe} = 17.71\ m/s$ safe operating limit established earlier. Figure 8 (d) shows that the *SR* system is able to improve safety while also maintaining the vehicle's capability to complete the mission.

TABLE III: Computational Latency Comparison

Module	*Average*	*Worst Case*
Mission Layer		
Segmentation	69362 μs	128634 μs
Fusion	922 μs	3665 μs
Safety Layer		
Obstacle Detection	7273 μs	38391 μs
Fault Handlers	387 μs	9801 μs

D. Computational Latency

A useful comparison between safety and mission layer components is their relative execution latency. The results are presented in Table III. The values only include the computational latency and do not include any time a module spends waiting for inputs.

Discussion. Safety layer components have significantly lower latency than those in the mission layer. This is a direct result of the reduced requirements that safety layer components need to fulfill. The low latency supports a faster response to external stimuli and the optimizations discussed in Section IV-B2.

VI. Discussion

Trade-offs. The improved safe operating capabilities of the AV come at the cost of degradation of comfort and performance, *e.g.,* due to reduced speed and increased braking. However, such impacts are direct results of hardware and software parameters. An advantage of using verifiable and logically analyzable software is that for each parameter the impact of different values can be determined analytically. For example, using a LiDAR with a larger sensor array will lead to higher safe speed limits (v_{max}^{safe}), albeit at additional sensor and compute hardware costs.

Verifiable Capabilities. The safety layer capabilities and commitments discussed in this work are limited. However, this is a limitation of the verification model of the algorithm [25] and not the SR architecture. As the verification model of the safety algorithms is expanded; reducing their constraints, reducing false positives while maintaining deterministic guarantees against false negatives and removing dependencies on obstacle properties; these improved verifiable capabilities can be integrated within SR.

Minimalistic Safety Layer. Safety layer components are designed to fulfill required tasks only, no optional features are included in the safety layer. This helps reduce the size of the safety layer software, simplifying software validation.

Platform Separation. As discussed in Section III-A, fault and temporal isolation between SR layers is an important requirement. In future works, we will study the platform separation of SR layers and the resultant challenges in real-time communication between the layers. The minimalist design of the safety layer software also allows it to run on low-cost embedded platforms, making it inexpensive to have hardware redundancy for the safety layer [64].

Planning and Control. In this work, we have focused on addressing perception faults first due to their role in real-world collisions. However, for end-to-end verifiable safety in AV, path planning and control also need to be addressed. In future works, we will evaluate existing safety solutions for planning and control for their suitability for use with SR.

Cost Reduction. SR not only improves safety in AV but also reduces costs. The deterministic guarantees simplify mission layer testing efforts and cost. It de-emphasizes the utility of hardware redundancy in the mission layer by making it safe to have a single mission layer copy, protected with highly redundant safety layers running on low-cost embedded platforms.

Incremental Design. SR allows incremental integration of fault handling capabilities, incrementally, but monotonically, improving the safety of the AV, while paving a path to end-to-end safety. As a consequence of this incremental design, safety overrides must be applied judiciously, only when a specific fault, handled within the safety layer, is detected. For example, consider a scenario when a deer suddenly jumps in front of the AV using the SR layer design for obstacle existence detection presented in this work. When the mission layer is unable to detect the existence of the deer, the safety layer overrides the control and applies brakes. This has multiple beneficial effects. The collision may be completely avoided or failing that the collision impact velocity is reduced. The time before the collision is increased, giving the mission layer more time to detect the deer and plan alternatives. However, if the mission layer does detect the existence of the deer, the safety layer removes its override, even if doing so leads to a collision with the deer. Within the presented design, the safety layer does not have the capability to determine the best possible path for the AV. A collision with deer can be the best response, avoiding more serious collisions with other vehicles. The safety layer, as presented in this work, does not have the capability to make this distinction and therefore does not intervene. However, as path planning capabilities are added to the safety layer, fast detections can be leveraged to initiate a collision-avoidance action with low latency (Table III).

Future Works. This paper presents the general design for SR and a reference design for obstacle existence detection faults. However, as shown by Figure 3, significant future research, developing verifiable capabilities, synergistic interactions, and integration within SR architecture, are required to achieve end-to-end verifiable safety in AV.

VII. Conclusion

This work presents Synergistic Redundancy (SR), a system architecture for verifiably safe complex cyber-physical systems. SR framework consists of a high-performance mission layer that is able to fulfill all functions of the system independently and a synergistic safety layer capable of keeping the system safe using verifiable software only. Synergistic cooperations between the layers further enhance their behaviors and therefore the whole system.

In autonomous vehicles (AV), safety in DNN-based perception is a critical challenge. In this work, the SR architecture is applied to AV towards a verifiable deterministic solution of obstacle existence detection faults, a crucial safety problem in AV. SR allows safe deployment of DNN with their incredible capabilities for perception in AV while protecting the system from infrequent, though critical, faults. While we directly address collision avoidance in the face of obstacle existence detection faults only, SR architecture is open to the inclusion of other fault tolerance capabilities, inviting future works.

Acknowledgment

The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant no. CNS 1932529, ECCS 2020289, the Air Force Office of Scientific Research (AFOSR) under grant no. #FA9550-21-1-0411, the National Aeronautics and Space Administration (NASA) under grant no. 80NSSC20M0229, AWD-000577-G1, and University of Illinois Urbana-Champaign under grant no. STII-21-06. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and don't necessarily reflect the views of the sponsors.

Authors would like to thank Prof. Tarek Abdelzaher, Dr. Micaela Verucchi, Jayati Singh, and Yitian Xue for their help, advice and guidance in the initial discussions for this work.

References

[1] T. Litman, *Autonomous vehicle implementation predictions, Implications for Transport Planning*. Victoria Transport Policy Institute Victoria, BC, Canada, 2022. [Online]. Available: https://www.vtpi.org/avip.pdf

[2] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.

[3] NVIDIA Corporation, "Hardware for Self-Driving Cars," 2022. [Online]. Available: https://www.nvidia.com/en-us/self-driving-cars/drive-platform/hardware/

[4] Fred Lambert, "Tesla unveils its new Full Self-Driving computer in detail: 'objectively the best chip in the world'," 2019. [Online]. Available: https://electrek.co/2019/04/22/tesla-full-self-driving-computer-details/

[5] SAE International, "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," https://www.sae.org/standards/content/j3016_201609, 2016.

[6] Coalition for Future Mobility, "Benefits of self driving vehicles," (accessed Dec 10, 2021). [Online]. Available: https://coalitionforfuturemobility.com/benefits-of-self-driving-vehicles/

[7] National Highway Traffic Safety Administration, "Automated vehicles for safety," (accessed Dec 10, 2021). [Online]. Available: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety

[8] M. Martínez-Díaz and F. Soriguera, "Autonomous vehicles: theoretical and practical challenges," *Transportation Research Procedia*, vol. 33, pp. 275–282, 2018.

[9] P. Koopman, "Autonomous vehicle myths: The dirty dozen," https://www.eetimes.com/autonomous-vehicle-myths-the-dirty-dozen/, 2021.

[10] D. Heaven *et al.*, "Why deep-learning ais are so easy to fool," *Nature*, vol. 574, no. 7777, pp. 163–166, 2019.

[11] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," *SAE International Journal of Transportation Safety*, vol. 4, no. 1, pp. 15–24, 2016.

[12] National Transportation Safety Board, "Collision Between a Car Operating With Automated Vehicle Control Systems and a Tractor-Semitrailer Truck Near Williston, Florida, May 7, 2016," Accident Report NTSB/HAR-17/02 PB2017-102600, September 2017. [Online]. Available: https://www.ntsb.gov/investigations/accidentreports/reports/har1702.pdf

[13] ——, "Collision between vehicle controlled by developmental automated driving system and pedestrian tempe, arizona, march 18, 2018," Accident Report NTSB/HAR-19/03 PB2019-101402, 2019 [Online]. [Online]. Available: https://www.ntsb.gov/investigations/AccidentReports/Reports/HAR1903.pdf

[14] ——, "Collision Between a Sport Utility Vehicle Operating With Partial Driving Automation and a Crash Attenuator Mountain View, California, March 23, 2018," Accident Report NTSB/HAR-20/01 PB2020-100112, February 2020. [Online]. Available: https://www.ntsb.gov/investigations/AccidentReports/Reports/HAR2001.pdf

[15] ——, "Highway accident brief HWY19FH008," Highway Accident Brief, Accident Number: HWY19FH008, March 2019. [Online]. Available: https://www.ntsb.gov/investigations/AccidentReports/Reports/HAB2001.pdf

[16] N. Altman, C. Weinstock, L. Sha, and D. Seto, "Simplex tm in a hostile communications environment: The coordinated prototype," Citeseer, Tech. Rep., 1999.

[17] L. Sha *et al.*, "Using simplicity to control complexity," *IEEE Software*, vol. 18, no. 4, pp. 20–28, 2001.

[18] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[19] Y. Zhang, X. Song, B. Bai, T. Xing, C. Liu, X. Gao, Z. Wang, Y. Wen, H. Liao, G. Zhang *et al.*, "2nd place solution for waymo open dataset challenge–real-time 2d object detection," *arXiv preprint arXiv:2106.08713*, 2021.

[20] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.

[21] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, "Pv-rcnn: Point-voxel feature set abstraction for 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 529–10 538.

[22] S. Shi, L. Jiang, J. Deng, Z. Wang, C. Guo, J. Shi, X. Wang, and H. Li, "Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection," *arXiv preprint arXiv:2102.00463*, 2021.

[23] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3d object detection and tracking," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 11 784–11 793.

[24] D. Fernandes, A. Silva, R. Névoa, C. Simões, D. Gonzalez, M. Guevara, P. Novais, J. Monteiro, and P. Melo-Pinto, "Point-cloud based 3d object detection and classification methods for self-driving applications: A survey and taxonomy," *Information Fusion*, vol. 68, pp. 161–191, 2021.

[25] A. Bansal, H. Kim, S. Yu, B. Li, N. Hovakimyan, M. Caccamo, and L. Sha, "Verifiable obstacle detection," *arXiv preprint arXiv:2208.14403*, 2022. [Online]. Available: https://arxiv.org/abs/2208.14403

[26] P. Koopman and M. Wagner, "Autonomous vehicle safety: An interdisciplinary challenge," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 1, pp. 90–96, 2017.

[27] C. Liu, T. Arnon, C. Lazarus, C. Strong, C. Barrett, and M. J. Kochenderfer, "Algorithms for verifying deep neural networks," *arXiv preprint arXiv:1903.06758*, 2019.

[28] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, and X. Yi, "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability," *Computer Science Review*, vol. 37, p. 100270, 2020.

[29] A. Albarghouthi, "Introduction to neural network verification," *arXiv preprint arXiv:2109.10317*, 2021.

[30] P. Feiler, J. Goodenough, A. Gurfinkel, C. Weinstock, and L. Wrage, "Four pillars for improving the quality of safety-critical software-reliant systems," Tech. Rep., 2013.

[31] M. Heimdahl, N. Leveson, J. Redler, M. Felton, and G. Lee, "Software assurance approaches, considerations, and limitations: Final report," Technical Report DOT/FAA/TC-15/57. Federal Aviation Administration, US DOT, Tech. Rep., 2016.

[32] C. Wilkinson, J. Lynch, R. Bharadwaj, and K. Woodham, "Verification of adaptive systems," Technical Report DOT/FAA/TC-16/4. Federal Aviation Administration, US DOT, Tech. Rep., 2016.

[33] E. Jenn, A. Albore, F. Mamalet, G. Flandin, C. Gabreau, H. Delseny, A. Gauffriau, H. Bonnin, L. Alecu, J. Pirard *et al.*, "Identifying challenges to the certification of machine learning for safety critical systems," in *European Congress on Embedded Real Time Systems (ERTS 2020)*, 2020.

[34] A. Pereira and C. Thomas, "Challenges of machine learning applied to safety-critical cyber-physical systems," *Machine Learning and Knowledge Extraction*, vol. 2, no. 4, pp. 579–602, 2020.

[35] Steven Posada, Office of Defects Investigation, NHTSA, USDOT, "ODI RESUME," Investigation PE 21-020, 2021 [Online]. [Online]. Available: https://static.nhtsa.gov/odi/inv/2021/INOA-PE21020-1893.PDF

[36] N. Kalra, *Challenges and approaches to realizing autonomous vehicle safety*. RAND Santa Monica, 2017.

[37] P. Penmetsa, P. Sheinidashtegol, A. Musaev, E. K. Adanu, and M. Hudnall, "Effects of the autonomous vehicle crashes on public perception of the technology," *IATSS Research*, 2021.

[38] W. Li, Y. Huang, S. Wang, and X. Xu, "Safety criticism and ethical dilemma of autonomous vehicles," *AI and Ethics*, pp. 1–6, 2022.

[39] A. R Yacoub and B. Briggs, "Liability and regulatory oversight of semi-autonomous and autonomous vehicles," *Available at SSRN: https://ssrn.com/abstract=4042306*, 2022.

[40] P. Jing, Y. Cai, B. Wang, B. Wang, J. Huang, C. Jiang, and C. Yang, "Listen to social media users: Mining chinese public perception of autonomous vehicles after crashes," *Available at SSRN: https://ssrn.com/abstract=4011917*, 2022.

[41] D. Ngo, "Dashcam shows fatal Tesla Model S crash in China," https://www.cnet.com/news/dash-cam-showed-fatal-tesla-crash-in-china, Sep 2016.

[42] F. Lambert, "Another fatal Tesla crash reportedly on Autopilot emerges, Model S hits a streetsweeper truck – caught on dashcam," https://electrek.co/2016/09/14/another-fatal-tesla-autopilot-crash-emerges-model-s-hits-a-streetsweeper-truck-caught-on-dashcam/, Sep 2016.

[43] Edward C. Chen and Joel Greer, "Umeda v. Tesla, Inc., United States District Court for the Northern District of California," Case No. 5:20-cv-2926: Docket No. 1, Complaint for damages, April 2020. [Online]. Available: https://regmedia.co.uk/2020/04/30/tesla_complaint.pdf

[44] N. Boudette, ""it happened so fast": Inside a fatal tesla autopilot accident," *The New York Times, Aug*, vol. 17, 2021.

[45] J. Torchinsky, "Tesla model 3 drives straight into overturned truck in what seems to be autopilot failure," *JALOPNIK*, (accessed June 01, 2020). [Online]. Available: https://jalopnik.com/tesla-model-3-drives-straight-into-overturned-truck-in-1843827507

[46] D. Kerman, "Lawsuit: Family blames tesla's autopilot for deadly crash," *KRON4*, 2021. [Online]. Available: https://www.kron4.com/news/bay-area/lawsuit-family-blames-teslas-autopilot-for-deadly-crash/

[47] J. Jernagan, "Fatal tesla crash into fire truck was one year ago today," *Banner Graphic*, 2020. [Online]. Available: https://www.bannergraphic.com/story/2856765.html

[48] AP News, "Feds will investigate deadly tesla crash in california," 2020. [Online]. Available: https://apnews.com/article/technology-business-los-angeles-us-news-california-6eae3986e7d9c1d00db7d52146cddf23

[49] G. RØED, "Tesla on auto-steering when man was cut down [translated]," *MOTOR*, 2020. [Online]. Available: https://www.motor.no/autopilot-nyheter-tesla/tesla-pa-auto-styring-da-mann-ble-meid-ned/188623

[50] C. Alexander, "Two dead in wreck involving tesla that pct. 4 says burned nearly four hours," *KHOU11*, 2021. [Online]. Available: https://www.khou.com/article/news/local/tesla-spring-crash-fire/285-c28a4993-5b5f-43f4-a924-e39638390647

[51] CBS Los Angeles, "Tesla driver killed, 2 seriously hurt in big-rig wreck on 210 freeway in fontana," 2021. [Online]. Available: https://www.cbsnews.com/losangeles/news/tesla-driver-killed-big-rig-wreck-on-210-freeway-in-fontana-good-samaritan-hurt/

[52] N. N. York, "52-year-old fixing flat on long island expressway hit, killed by tesla driver: Cops," 2021. [Online]. Available: https://www.nbcnewyork.com/news/local/52-year-old-fixing-flat-on-long-island-expressway-hit-killed-by-tesla-driver-cops/3175414/

[53] Baidu, "Apollo," http://apollo.auto/, 202.

[54] I. Bogoslavskyi and C. Stachniss, "Fast range image-based segmentation of sparse 3d laser scans for online operation," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 163–169.

[55] ——, "Efficient online segmentation for sparse 3d laser scans," *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, vol. 85, no. 1, pp. 41–52, 2017.

[56] LG Silicon Valley Lab, "LG Silicon Valley Lab Apollo 5.0 Fork," https://github.com/lgsvl/apollo-5.0, 2019.

[57] LG Electronics America R&D Center Advanced Platform Lab, "LGSVL Simulator ," https://www.lgsvlsimulator.com/, 2020.

[58] K. Ishii, A. Noguchi, and Y. Gotoh, "Fault tolerable redundancy control," Apr. 15 1986, US Patent 4,583,224.

[59] J. H. Lala and R. E. Harper, "Architectural principles for safety-critical real-time applications," *Proceedings of the IEEE*, vol. 82, no. 1, pp. 25–40, 1994.

[60] R. Hammett, "Design by extrapolation: an evaluation of fault-tolerant avionics," in *20th DASC. 20th Digital Avionics Systems Conference (Cat. No.01CH37219)*, vol. 1, Oct 2001, pp. 1C5/1–1C5/12 vol.1.

[61] R. Isermann, R. Schwarz, and S. Stolzl, "Fault-tolerant drive-by-wire systems," *IEEE Control Systems Magazine*, vol. 22, no. 5, pp. 64–81, 2002.

[62] M. Baleani, A. Ferrari, L. Mangeruca, A. Sangiovanni-Vincentelli, M. Peri, and S. Pezzini, "Fault-tolerant platforms for automotive safety-critical applications," in *Proceedings of the 2003 international conference on Compilers, architecture and synthesis for embedded systems*. ACM, 2003, pp. 170–177.

[63] R. E. Bickel, "Fault tolerant processing architecture," Aug. 30 2005, US Patent 6,938,183.

[64] H. Kim, H. Lee, and K. Lee, "The design and analysis of avtmr (all voting triple modular redundancy) and dual–duplex system," *Reliability Engineering & System Safety*, vol. 88, no. 3, pp. 291–300, 2005.

[65] O. Rooks, M. Armbruster, A. Sulzmann, G. Spiegelberg, and U. Kiencke, "Duo duplex drive-by-wire computer system," *Reliability Engineering & System Safety*, vol. 89, no. 1, pp. 71–80, 2005.

[66] J. R. Marshall and D. G. Langston, "Error detection and fault isolation for lockstep processor systems," May 16 2000, US Patent 6,065,135.

[67] H. Oyamada and M. Shiga, "Multi-processor system with fault detection," Sep. 19 1995, US Patent 5,452,443.

[68] M. Rebaudengo, M. Sonza Reorda, M. Torchiano, and M. Violante, "Soft-error detection through software fault-tolerance techniques," in *Proceedings 1999 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (EFT'99)*, Nov 1999, pp. 210–218.

[69] A. Mahmoud, S. K. S. Hari, M. B. Sullivan, T. Tsai, and S. W. Keckler, "Optimizing software-directed instruction replication for gpu error detection," in *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2018, pp. 842–853.

[70] A. Runge, "Reliability enhancement of fault-prone many-core systems combining spatial and temporal redundancy," in *2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems*. IEEE, 2012, pp. 1600–1605.

[71] J. Wei, J. M. Snider, J. Kim, J. M. Dolan, R. Rajkumar, and B. Litkouhi, "Towards a viable autonomous driving research platform," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, June 2013, pp. 763–770.

[72] P. Koopman and M. Wagner, "Autonomous vehicle safety: An interdisciplinary challenge," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 1, pp. 90–96, Spring 2017.

[73] S. Davies, "Methods and apparatus for processor system having fault tolerance," Feb. 8 2007, US Patent App. 11/198,198.

[74] N. Wiersma and R. Pareja, "Safety!= security: On the resilience of asil-d certified microcontrollers against fault injection attacks," in *2017 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*. IEEE, 2017, pp. 9–16.

[75] S. Jha, S. Banerjee, T. Tsai, S. K. Hari, M. B. Sullivan, Z. T. Kalbarczyk, S. W. Keckler, and R. K. Iyer, "Ml-based fault injection for autonomous vehicles: A case for bayesian fault injection," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2019, pp. 112–124.

[76] M. Blanke, M. Kinnaert, J. Lunze, M. Staroswiecki, and J. Schröder, *Diagnosis and fault-tolerant control*. Springer, 2006, vol. 2.

[77] D. Seto, B. H. Krogh, L. Sha, and A. Chutinan, "Dynamic control system upgrade using the simplex architecture," *IEEE Control Systems Magazine*, vol. 18, no. 4, pp. 72–80, 1998.

[78] T. L. Crenshaw, E. L. Gunter, C. L. Robinson, L. Sha, and P. R. Kumar, "The simplex reference model: Limiting fault-propagation due to unreliable components in cyber-physical system architectures," in *Proceedings of the 28th IEEE Real-Time Systems Symposium (RTSS 2007), 3-6 December 2007, Tucson, Arizona, USA*, 2007, pp. 400–412. [Online]. Available: https://doi.org/10.1109/RTSS.2007.34

[79] S. Bak, D. K. Chivukula, O. Adekunle, M. Sun, M. Caccamo, and L. Sha, "The system-level simplex architecture for improved real-time embedded system safety," in *15th IEEE Real-Time and Embedded Technology and Applications Symposium*, San Francisco, USA, 2009, pp. 99–107.

[80] X. Wang, N. Hovakimyan, and L. Sha, "Rsimplex: A robust control architecture for cyber and physical failures," *ACM Transactions on Cyber-Physical Systems*, vol. 2, no. 4, pp. 1–26, 2018.

[81] Y. Mao, Y. Gu, N. Hovakimyan, L. Sha, and P. Voulgaris, "Sl1-simplex: Safe velocity regulation of self-driving vehicles in dynamic and unforeseen environments," *arXiv preprint arXiv:2008.01627*, 2020.

[82] P. Musau, N. Hamilton, D. M. Lopez, P. Robinette, and T. T. Johnson, "On using real-time reachability for the safety assurance of machine learning controllers," in *2022 IEEE International Conference on Assured Autonomy (ICAA)*. IEEE, 2022, pp. 1–10.

[83] D. T. Phan, R. Grosu, N. Jansen, N. Paoletti, S. A. Smolka, and S. D. Stoller, "Neural simplex architecture," in *NASA Formal Methods Symposium*. Springer, 2020, pp. 97–114.

[84] P. Burgio, M. Bertogna, N. Capodieci, R. Cavicchioli, M. Sojka, P. Houdek, A. Marongiu, P. Gai, C. Scordino, and B. Morelli, "A software stack for next-generation automotive systems on many-core heterogeneous platforms," *Microprocessors and Microsystems*, vol. 52, pp. 299–311, 2017.

[85] Nvidia Corporation, "Nvidia drive os," https://developer.nvidia.com/drive/drive-software#driveav, 2020.

[86] R. Mirosanlou, M. Hassan, and R. Pellizzoni, "Duetto: Latency guarantees at minimal performance cost," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021, pp. 1136–1141.

[87] Y. Yuan, R. Tasik, S. S. Adhatarao, Y. Yuan, Z. Liu, and X. Fu, "Race: reinforced cooperative autonomous vehicle collision avoidance," *IEEE Transactions on Vehicular Technology*, 2020.

[88] K. Lee and D. Kum, "Collision avoidance/mitigation system: Motion planning of autonomous vehicle via predictive occupancy map," *IEEE Access*, vol. 7, pp. 52 846–52 857, 2019.

[89] H. Wang, Y. Huang, A. Khajepour, Y. Zhang, Y. Rasekhipour, and D. Cao, "Crash mitigation in motion planning for autonomous vehicles," *IEEE transactions on intelligent transportation systems*, vol. 20, no. 9, pp. 3313–3323, 2019.

[90] H. Tahir, M. N. Syed, and U. Baroudi, "Heuristic approach for real-time multi-agent trajectory planning under uncertainty," *IEEE Access*, vol. 8, pp. 3812–3826, 2019.

[91] S. Noh and K. An, "Decision-making framework for automated driving in highway environments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 58–71, 2017.

[92] S. Noh, “Decision-making framework for autonomous driving at road intersections: Safeguarding against collision, overly conservative behavior, and violation vehicles,” *IEEE Transactions on Industrial Electronics*, vol. 66, no. 4, pp. 3275–3286, 2018.

[93] D. Shin, B. Kim, K. Yi, A. Carvalho, and F. Borrelli, “Human-centered risk assessment of an automated vehicle using vehicular wireless communication,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 2, pp. 667–681, 2018.

[94] M.-Y. Yu, R. Vasudevan, and M. Johnson-Roberson, “Occlusion-aware risk assessment for autonomous driving in urban environments,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2235–2241, 2019.

[95] G. Li, Y. Yang, T. Zhang, X. Qu, D. Cao, B. Cheng, and K. Li, “Risk assessment based collision avoidance decision-making for autonomous vehicles in multi-scenarios,” *Transportation research part C: emerging technologies*, vol. 122, p. 102820, 2021.

[96] P. Seiler, B. Song, and J. K. Hedrick, “Development of a collision avoidance system,” *SAE transactions*, pp. 1334–1340, 1998.

[97] J. Funke, M. Brown, S. M. Erlien, and J. C. Gerdes, “Collision avoidance and stabilization for autonomous vehicles in emergency scenarios,” *IEEE Transactions on Control Systems Technology*, vol. 25, no. 4, pp. 1204–1216, 2016.

[98] X. He, Y. Liu, C. Lv, X. Ji, and Y. Liu, “Emergency steering control of autonomous vehicle for collision avoidance and stabilisation,” *Vehicle system dynamics*, vol. 57, no. 8, pp. 1163–1187, 2019.

[99] S. Cheng, L. Li, H.-Q. Guo, Z.-G. Chen, and P. Song, “Longitudinal collision avoidance and lateral stability adaptive control system based on mpc of autonomous vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 6, pp. 2376–2385, 2019.

[100] R. Hajiloo, M. Abroshan, A. Khajepour, A. Kasaiezadeh, and S.-K. Chen, “Integrated steering and differential braking for emergency collision avoidance in autonomous vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[101] M. Realpe, B. X. Vintimilla, and L. Vlacic, “Multi-sensor fusion module in a fault tolerant perception system for autonomous vehicles,” in *2nd International Conference on Robotics and Artificial Intelligence, Los Angeles, USA*, 2016.

[102] ——, “A fault tolerant perception system for autonomous vehicles,” in *2016 35th Chinese Control Conference (CCC)*. IEEE, 2016, pp. 6531–6536.

[103] D. Jackson, V. Richmond, M. Wang, J. Chow, U. Guajardo, S. Kong, S. Campos, G. Litt, and N. Arechiga, “Certified control: An architecture for verifiable safety of autonomous vehicles,” *arXiv preprint arXiv:2104.06178*, 2021.

[104] S. Xi, J. Wilson, C. Lu, and C. Gill, “Rt-xen: Towards real-time hypervisor scheduling in xen,” in *2011 Proceedings of the Ninth ACM International Conference on Embedded Software (EMSOFT)*. IEEE, 2011, pp. 39–48.

[105] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, and T. Azumi, “Autoware on board: Enabling autonomous vehicles with embedded systems,” in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2018, pp. 287–296.

[106] W.-S. Lee, D. L. Grosh, F. A. Tillman, and C. H. Lie, “Fault tree analysis, methods, and applications: A review,” *IEEE Transactions on Reliability*, vol. 34, no. 3, pp. 194–203, 1985.

[107] V. Schönemann, H. Winner, T. Glock, E. Sax, B. Boeddeker, S. vom Dorff, G. Verhaeg, F. Tronci, and G. G. Padilla, “Fault tree-based derivation of safety requirements for automated driving on the example of cooperative valet parking,” in *26th International Technical Conference on The Enhanced Safety of Vehicles (ESV)*, vol. 9, 2019, p. 2019.

[108] P. Bhavsar, P. Das, M. Paugh, K. Dey, and M. Chowdhury, “Risk analysis of autonomous vehicles in mixed traffic streams,” *Transportation Research Record*, vol. 2625, no. 1, pp. 51–61, 2017.

[109] B. Wan, H. Luo, K. Zhou, X. Li, C. Wang, X. Chen, and X. Zhou, “A time-aware programming framework for constructing predictable real-time systems,” in *2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 2017, pp. 578–585.

[110] C. Menard, A. Goens, M. Lohstroh, and J. Castrillon, “Achieving determinism in adaptive autosar,” in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 822–827.

[111] W. Baron, A. Arestova, C. Sippl, K.-S. Hielscher, and R. German, “Lett: An execution model for distributed real-time systems,” in *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*. IEEE, 2021, pp. 1–7.

[112] Baidu, “Apollo Cyber RT,” https://github.com/ApolloAuto/apollo/tree/master/cyber, 2022.

[113] S. Liu, S. Yao, X. Fu, R. Tabish, S. Yu, A. Bansal, H. Yun, L. Sha, and T. Abdelzaher, “On removing algorithmic priority inversion from mission-critical machine inference pipelines,” in *2020 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2020, pp. 319–332.

[114] S. Liu, S. Yao, X. Fu, H. Shao, R. Tabish, S. Yu, A. Bansal, H. Yun, L. Sha, and T. Abdelzaher, “Real-time task scheduling for machine perception in in intelligent cyber-physical systems,” *IEEE Transactions on Computers*, 2021.

[115] Y. Hu, S. Liu, T. Abdelzaher, M. Wigness, and P. David, “On exploring image resizing for optimizing criticality-based machine perception,” in *2021 IEEE 27th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. IEEE, 2021, pp. 169–178.

[116] ——, “Real-time task scheduling with image resizing for criticality-based machine perception,” *Real-Time Systems*, pp. 1–26, 2022.

[117] J. Chen, S. Yu, R. Tabish, A. Bansal, S. Liu, T. Abdelzaher, and L. Sha, “Lidar cluster first and camera inference later: A new perspective towards autonomous driving,” *arXiv preprint arXiv:2111.09799*, 2021.

[118] W. Kang, S. Chung, J. Y. Kim, Y. Lee, K. Lee, J. Lee, K. G. Shin, and H. S. Chwa, “Dnn-sam: Split-and-merge dnn execution for real-time object detection,” in *2022 IEEE 28th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2022, pp. 160–172.

[119] S. Liu, X. Fu, M. Wigness, P. David, S. Yao, L. Sha, and T. Abdelzaher, “Self-cueing real-time attention scheduling in criticality-aware visual machine perception,” in *2022 IEEE 28th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2022, pp. 173–186.

[120] LG Electronics America R&D Center Advanced Platform Lab, “Lidar Sensor Plugin,” https://www.svlsimulator.com/docs/archive/2020.06/lidar-plugin/, 2020.

[121] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine *et al.*, “Scalability in perception for autonomous driving: Waymo open dataset,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2446–2454.

[122] M. Bijelic, T. Gruber, and W. Ritter, “A benchmark for lidar sensors in fog: Is detection breaking down?” in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 760–767.

[123] R. Heinzler, P. Schindler, J. Seekircher, W. Ritter, and W. Stork, “Weather influence and classification with automotive lidar sensors,” in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 1527–1534.

[124] A. M. Wallace, A. Halimi, and G. S. Buller, “Full waveform lidar for adverse weather conditions,” *IEEE transactions on vehicular technology*, vol. 69, no. 7, pp. 7064–7077, 2020.

[125] Y. Li, P. Duthon, M. Colomb, and J. Ibanez-Guzman, “What happens for a tof lidar in fog?” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 11, pp. 6670–6681, 2020.

[126] M. Pavlić, H. Belzner, G. Rigoll, and S. Ilić, “Image based fog detection in vehicles,” in *2012 IEEE Intelligent Vehicles Symposium*. IEEE, 2012, pp. 1132–1137.

[127] P. Sallis, C. Dannheim, C. Icking, and M. Maeder, “Air pollution and fog detection through vehicular sensors,” in *2014 8th Asia Modelling Symposium*. IEEE, 2014, pp. 181–186.

[128] D. J. Straub *et al.*, “Detecting the presence of fog using low-cost proximity sensors,” *Aerosol and Air Quality Research*, vol. 20, no. 5, pp. 981–990, 2020.

[129] K. Larson, “Dust detection and warning system tracks its first season,” https://azdot.gov/adot-blog/dust-detection-and-warning-system-tracks-its-first-season, Arizona Department of Transportation, 2020.

[130] R.-C. Miclea, V.-I. Ungureanu, F.-D. Sandru, and I. Silea, “Visibility enhancement and fog detection: Solutions presented in recent scientific papers with potential for application to mobile systems,” *Sensors*, vol. 21, no. 10, p. 3370, 2021.

[131] J. Petit, B. Stottelaar, M. Feiri, and F. Kargl, “Remote attacks on automated vehicles sensors: Experiments on camera and lidar,” *Black Hat Europe*, vol. 11, no. 2015, p. 995, 2015.

[132] H. Shin, D. Kim, Y. Kwon, and Y. Kim, "Illusion and dazzle: Adversarial optical channel exploits against lidars for automotive applications," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2017, pp. 445–467.

[133] Y. Cao, C. Xiao, D. Yang, J. Fang, R. Yang, M. Liu, and B. Li, "Adversarial objects against lidar-based autonomous driving systems," *arXiv preprint arXiv:1907.05418*, 2019.

[134] J. Sun, Y. Cao, Q. A. Chen, and Z. M. Mao, "Towards robust {LiDAR-based} perception in autonomous driving: General black-box adversarial sensor attack and countermeasures," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 877–894.

[135] H. Weichel, *Laser beam propagation in the atmosphere*. SPIE press, 1990, vol. 10319.

[136] S. Kaasalainen, H. Hyyppa, A. Kukko, P. Litkey, E. Ahokas, J. Hyyppa, H. Lehner, A. Jaakkola, J. Suomalainen, A. Akujarvi *et al.*, "Radiometric calibration of lidar intensity with commercially available reference targets," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 2, pp. 588–598, 2008.

[137] I. I. Kim, B. McArthur, and E. J. Korevaar, "Comparison of laser beam propagation at 785 nm and 1550 nm in fog and haze for optical wireless communications," in *Optical wireless communications III*, vol. 4214. Spie, 2001, pp. 26–37.

[138] R.-C. Miclea, C. Dughir, F. Alexa, F. Sandru, and I. Silea, "Laser and lidar in a system for visibility distance estimation in fog conditions," *Sensors*, vol. 20, no. 21, p. 6322, 2020.

[139] A. Bansal, J. Singh, M. Verucchi, M. Caccamo, and L. Sha, "Risk ranked recall: Collision safety metric for object detection systems in autonomous vehicles," in *2021 10th Mediterranean Conference on Embedded Computing (MECO)*, 2021.

[140] C. Schmidt, F. Oechsle, and W. Branz, "Research on trajectory planning in emergency situations with multiple objects," in *2006 IEEE Intelligent Transportation Systems Conference*. IEEE, 2006, pp. 988–992.

[141] Baidu, "Apollo 3.5 Software Architecture," https://github.com/ApolloAuto/apollo/blob/master/docs/specs/Apollo_3.5_Software_Architecture.md, 2020.

[142] LG Silicon Valley Lab, "lgsvl - A Python API for SVL Simulator," https://github.com/lgsvl/PythonAPI, 2021.

[143] ——, "Lincoln MKZ 2017," https://content.lgsvlsimulator.com/vehicles/lincolnmkz2017/, 2021.